Optimization of 3-D VLSI Cell Placement for Reduction in TSVs and Area

BY

MOHAMMED AHMAD KHALIL

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

March 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Mohammed Ahmad Abdel Rahman Khalil** under the direction his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING.**

Sadiq Sait. M

Dr. Sadiq M. Sait
(Advisor)

Dr. Ahmad Al-Mulhem
Department Chairman

Alaa Amin

Dr. Alaaeldin Amin
(Member)

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

Dr. Shokri Selim
(Member)

22/3/15

Date

Dedication

To my beloved mother

ACKNOWLEDGMENTS

First of all, infinite thanks and complete gratitude to almighty Allah for his unstoppable blessings. Many thanks, to my father, my mother, and my whole family for their trust and support through this work as well as through all of my life.

Special thanks to my mentor and advisor Professor Sadiq M. Sait for his appreciated guidance and directions. Thanks to my committee members: Dr. Alaaeldin Amin and Dr. Shokri Selim for their motivation and valuable suggestions.

Dedicated thanks to Mr. Feras Oughli and Mr. Mohammed Al-Asli for helping me through this thesis. Thanks to COE department and KFUPM for giving me this opportunity. Finally, special thanks are due to my senior colleagues at KFUPM for their help and prayers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IC	:	Integrated Circuit
2D	:	Two Dimensional
3D	:	Three Dimensional
VLSI	:	Very Large Scale Integrated Circuit
TSV	:	Through Silicon Via
WL	:	Wire Length
IO	:	Input-output
SimE	:	Simulated Evolution
FD	:	Force Directed
SA	:	Simulated Annealing
TS	:	Tabu Search
WSR	:	White Space Redistribution
MST	:	Minimum Spanning Tree
TTS	:	Thermal Through Silicon
LP	:	Linear Programming
ILP	:	Integer Linear Programming
NLP	:	Nonlinear Programming Problem
MCMF	:	Minimum Cost Maximum Flow
PSP	:	Partitioned Sequence Pairs

CBL	:	Corner Block List
CBA	:	Combined Bucket Array
AR	:	Aspect Ratio
SP	:	Semi Perimeter
SD	:	Standard Deviation
COV	:	Coefficient of Variance
TCO	:	Total Combined Objective
GBSA	:	Goodness Based Simulated Annealing
FM	:	Fiduccia-Mathyeses
TCG	:	Total Combined Goodness

ABSTRACT

Full Name : Mohammed Ahmad Abdel Rahman Khalil

Thesis Title : Optimization of 3-D VLSI Cell Placement for Reduction in TSVs and Area

Major Field : Computer Engineering

Date of Degree : March.2015

3-Dimensional Integrated Circuit is a new technology that overcomes many of the 2-Dimensional Integrated Circuit technology limitations especially the long interconnects delay and the large flat area. In this technology multiple dies are stacked vertically leading to a reduced flat area. This technology utilizes Through Silicon Vias (TSVs) as interconnects for nets that span multiple layers. Like any other technology, 3-Dimensional Integrated Circuit comes with its own problems and challenges especially in the physical design process. This work investigated the main challenges that are imposed on this new technology and solved two of the important physical design problems using evolutionary iterative heuristics. The first problem is the TSV minimization problem under the area balancing constraint. In this work we deployed and adapted a modified Simulating annealing algorithm and the simulated evolution algorithm to solve the TSV optimization problem. The results show that the modified Simulating annealing algorithm outperforms the standard Simulating annealing and the simulated evolution algorithm outperforms both approaches along with the Tabu search algorithm. The second problem we handled in this work is the placement problem in 3D cell design. In this problem the objective is to minimize both of the number of TSVs and the overall wirelength. To solve this hard problem, we applied a simulated evolution algorithm that incorporates the force directed algorithm in its allocation step; the results show high quality solutions in term of the TSV Count and the total Wirelength.

خلاصة الرسالة

الاسم الكامل: محمد احمد عبد الرحمن خليل

عنوان الرسالة: تحسين مسألة الوضع لتصاميم الخلية ثلاثية الأبعاد بهدف تقليل عدد الدسر العابرة للسليكون والمساحة الكلية

التخصص: هندسة الحاسب الألي

تاريخ الدرجة العلمية: ديسمبر 2014

تتغلب تقنية الدوائر المتكاملة ثلاثية الأبعاد الجديدة على العديد من تحديات سابقتها ثنائية الأبعاد، وخاصة فيما يتعلق بالتأخر الناتج من التوصيلات الطويلة وما يتعلق بكبر المساحات المسطحة، حيث يصبح من الممكن تكديس العديد من الخرد مما يؤدي إلى تقليص المناطق المسطحة. تستخدم هذه التقنية دسرا عابرة للسليكون لتوصيل الشبكات التي تغطي طبقات متعددة. ومثل أي تقنية أخرى، فإن للدوائر المتكاملة ثلاثية الأبعاد مشاكلها وتحدياتها الخاصة لا سيما فيما يتعلق بعملية التصميم المادي. بحث هذا العمل في التحديات الرئيسية التي تفرض نفسها على هذه التقنية الجديدة وحلت اثنتين من مشاكل تصميمها المادي المهمة باستخدام المُلح التطورية التكرارية (Evolutionary Iterative Heuristics). أما المشكلة الأولى فتتعلق بمسألة تقليص الدسر العابرة للسليكون تحت قيد المنطقة المتوازنة. في هذا العمل قمنا بتطويع خوارزمية معدلة من خوارزميات محاكاة الانصهار(Simulating Annealing Algorithm) وكيفناها لحل لإيجاد قيم المسألة المثلى; وقد أظهرت النتائج أن النسخة المعدلة من الخوارزمية تفوقت على نظيرتها القياسية كما تفوقت خوارزمية المحاكاة التطورية (Simulated Evolution Algorithm) كلا النهجين جنبا إلى جنب مع خوارزمية تابو للبحث (Tabu Search Algorithm). وأما المشكلة الثانية التي تعاملنا معها في هذا العمل فتكمن في مسألة الوضع لتصاميم الخلية ثلاثية الأبعاد، حيث يتمثل الهدف في تقليل كل من عدد الدسر العابرة للسليكون وأطوال الأسلاك إلى أدنى حد. ولحل هذه المشكلة العويصة، فقد قمنا بتطبيق خوارزمية محاكاة تطورية تضمنت خوارزمية القوة الموجهة (Force Directed Algorithm) في خطوة الوضع. وقد إظهرت النتائج حلولا عالية الجودة لكل من أعداد الدسر العابرة للسليكون وأطوال الأسلاك على حد السواء.

CHAPTER 1

INTRODUCTION

Throughout the last decade, the evolving of the high performance computing and hardware based applications pushed the semiconductor technology to pass through many advancements in many aspects. One of the most important and significant recent improvements in this field is the decreasing of transistor's feature size; this enhancement increases the IC (Integrated Circuit) maximum number of transistors and consequently minimizing the chip area while improving yield. However the higher IC transistor density boosts some design, performance and reliability issues like power dissipation and leakage, signal integrity, yield degradation and clock issues [1]. Moreover, the non-scalable interconnect delay in any IC is progressively controlling its performance [1]. To mitigate these difficulties, overcome these considerable issues and facilitate the manufacturing process, some solutions have been explored. Due to the physical limitations, the typical 2D (two dimensional) scaling methods failed and 3D (three dimensional) approaches have been suggested and utilized [2-4].

In 3D VLSI (Very Large Scale Integrated Circuit) technology, a number of ICs are stacked on a chip to achieve higher integration, lower power consumption, smaller footprint, and improved performance (reduced wirelength (WL)) [5]. Unlike traditional 2D IC designs where the modules are distributed over a one layer 2D chip, 3D technology stacked dies in vertical tiers (aka layers), which requires new methods to vertically interconnect these tiers. A lot of studies and researches recommend the through-silicon via (TSV) as a powerful and promising method for vertical interconnection between the different layers in 3D VLSI designs [4].

1.1 3-Dimentional Integrated Circuits

As mentioned earlier, 3D technology dominated the 2D technology by providing higher chip density and higher onboard bandwidth [6]; this technology has a strong

impact on the reduction of the chip area and it replaces the long required interconnecting wires by a short ones; thus improves the system performance significantly. Also in this technology, modules with various power, voltage and performance properties can be distributed among different layers to accommodate its requirements [2]. The key idea in 3D technology is to stack multiple smaller dies vertically while connecting those chips using any 3D vias such as the TSVs. Another important issue in this technology is the handling of the bonding process which has been carried by two main techniques: wafer-2-wafer and die-2-wafer methods [4]. Wafer-2-wafer bonding technique is suitable for high yielding tiers that consist of the same size die. On the other hand, die-2-wafer method is more practical for varying size dies and low yielding tiers [4].

Regular chips normally can be viewed as a stack of three Layers: the silicon substrate, the device layer, and the metal layer as shown in Figure 1.1 [7]. Stacking and aligning these 2D chips to form 3D chip is an important and sensitive process in 3D IC technology that handled mainly by one of the following approaches: Face-2-Face (F2F), Back-2-Back (B2B), and Face-2-Back (F2B) stacking styles.

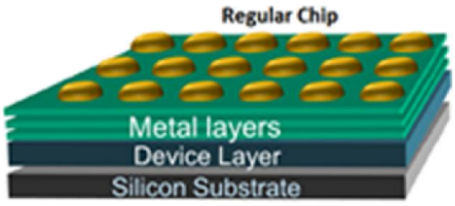

Figure 1.1: Layers of Regular 2D Chip [7].

In F2F style, only 2 layers can be stacked where the front pads of both chips are directly connected to each other within the bonding interface [8].

In order to stack three or more layers, the other stacking styles can be used along with TSV technology. B2B style uses two TSVs (one in each layer) to achieve the connection between two neighbor layers which leads to increase the number of TSVs required (area increasing) and introduces longer delays. F2B is the commonly used process but it requires an extra processing step and backsides metal layers to connect the

front of one layer to the back of the other one. Figure 1.2 shows the extra required metal layers for the F2B style [7].

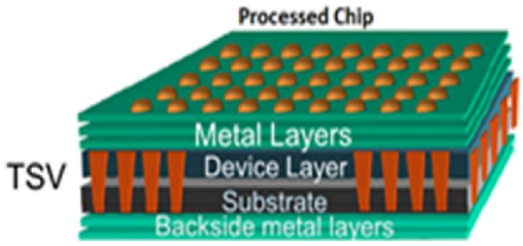

Figure 1.2: Layers of The Processed 2D Chip in F2B style [7].

From the above explanation of the general nature of the 3D IC technology we can observe that it is a promising technology that requires special manufacturing and fabrication processes and techniques. To optimally exploit this new technology, a lot of work needs to be done on the physical design side to optimize this VLSI technology in terms of floorplanning, placement and routing.

1.1.1 Through Silicon Vias

TSV is the most promising and powerful solution for layers vertical interconnection in 3D Chips. TSV could be viewed as a 3D via or metal channel that connects inter-tiers net (a group of electrically connected pins) by permeating through any available white space in the silicon substrate [9]. TSV is fabricated mainly using one of the following approaches: via-first and via-last [10]. In via-first approach the TSV has typically 1 to 5 µm diameter, it penetrates through the device layer only, and so it can be treated as a special via that can be placed within the chip under some special considerations [11, 12]. Via-last TSV has typically 5 to 20 µm diameters, it goes through the device and metal layers and accordingly it is usually modeled as a cell that has limited placement locations [13]. Figure 1.3 explains those TSV fabrication methods in F2B stacking style.

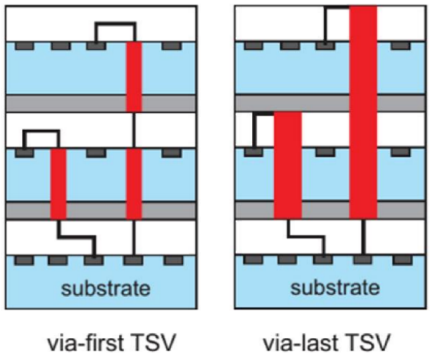

Figure 1.3: TSV Fabrication Methods: Via-first and Via-last [13].

Modules belong to the same net and lie on different layers are connected by a TSV. To achieve the connection between the modules and TSV, metal landing pads are usually used, as explained in Figure 1.4 [8]. These pads are developed on the top metal layer, and since each layer is fabricated independently and post alignment process of different tiers is usually carried, the size of these pads has to be large enough to overcome any possibilities of misalignment [14].

Figure 1.4: TSV with Metal Landing Pad [8].

Along with its usefulness and powerful role in 3D IC technology, TSV come with its own limitations and it introduces extra significant problems that need to be handled carefully in 3D IC designs. TSV size and intra-TSV spacing are examples of these problems.

Compared to the size of gates and memory cells, TSV size is larger by several times; [7] stated that in a 32-nm technology the TSV dimension is larger than the standard cell height by at least 5 times. This large TSV size means that the number of used TSVs is a key factor that impacts significantly the required design area. Also since

TSV have to be placed somewhere between the modules, the routing process will be affected by these TSVs and consequently the package area will increase.

Another important TSV issue is the spacing between the TSVs themselves and between the TSVs and the modules. Fabricated TSVs suffer from a mechanical stress around them which requires TSVs to be placed apart in order to improve the resultant yield. Another important spacing issue is to preserve a keep-out zone around TSV where the modules shouldn't be placed in order to avoid the effect of the thermo-mechanical stress on the nearby modules carrier mobility [5]. This extra spacing obviously will increase the required area if the 3D physical design is not handled properly.

The earlier mentioned negative impact of TSV on 3D design highlights the need to handle 3D VLSI physical design at all stages in a very thorough manner, considering the characteristics and effects of these TSVs on the performance and cost of the design. These issues attracted the attention of many researchers during the past years and accordingly many solutions, approaches and methods for TSV minimization, total area reduction and thermal limitations in the floorplanning and placement problems of the 3D physical design process have been proposed, explored, implemented, and evaluated.

1.2 Physical Design in VLSI

In VLSI technology one of the most important phases that come before fabricating circuits and chips is the circuit physical design stage. After the logic design of any circuit is carried out by digital design specialists and engineers, the physical design phase takes place to handle and complete all required synthesis steps. This stage includes partitioning, floorplanning, placement and routing steps. The importance of this stage is due to its sensitive and great impact on many of the circuit desired characteristics such as the circuit performance, reliability, area, and yield. Placement and routing are the dominant steps that determine the length of the interconnecting wires in the circuit; this means that if these steps are not handled intelligently, then the interconnecting wires will be longer and consequently the timing performance of the chip will degrade considerably. Furthermore, in any IC there is two main parts that contribute to the overall design area: the modules area which has nothing to do with the physical design, and the wiring area.

Since the amount of required wiring is directly related to the way the modules are placed and distributed; the placement will play a major role in determining the total design area. Also increasing the area will decrease the yield and accordingly raise the production cost, which again highlights the major effect of the physical design on the overall VLSI design process [15].

The importance of VLSI physical design has catalyzed the researches to focus on a smart and effective ways that can be developed to optimize all the mentioned steps and thus enhancing the fabricated chips while minimizing the corresponding production cost. For the regular 2D IC all the physical design steps are performed on a normal 2D plane; the floorplanning and placement problems involve only a set of modules that need to be distributed on a 2D surface; the routing stage defines the paths on the very same planar surface. On the other hand 3D IC physical design deal with a multi-layer chips that are connected through TSVs. In this technology, all of the discussed steps are more complicated and require more intelligent and efficient approaches. The placement problem for example involves the distributed of modules on different layers as well as on every 2D tier. Also since the TSV has a significant area as mentioned earlier, the placement approaches should take into account the distributing of these TSVs on their corresponding layers. However, since 3D chips can be viewed as a number of parallel 2D dies, the 2D physical design approaches can be modified and scaled to handle some of the steps in the 3D technology.

Both 2D and 3D physical design problems are hard problems that require developing and applying of suitable and effective optimization approaches. Our work focuses on solving one of these problems which is the placement problem in 3D cells Design. In cells design each die can be viewed as an array of rows where each row consists of a number of cells where exactly one module or TSV can be placed. In regular 2D cells the objective of this problem is to minimize the total WL that interconnects the placed cells. But in 3D chips the problem has minimizing the number of required TSVs as a second objective. To solve the discussed problem our work employs one of the iterative algorithms (Simulated Evolution) that incorporates another numerical (Force Directed) and deterministic algorithms. Since this problem is a multi-objective

optimization problem, the adapted solution is applying the fuzzy logic to handle this issue.

1.3 Evolutionary Algorithms

Optimization problems are one of the most important issues that appeared in different disciplines and in a variety of designs and applications. Because of its popularity, this class of problems have been discussed and addressed by many researches during the past decades. In every optimization problem the main target is to maximize or minimize some objective function or in other words, finding the best solution.

Mainly the optimization problem is carried out using one of the following approaches: exact algorithms and approximation algorithms (aka Heuristics). Exact algorithms are deterministic and produce the final exact best solution. For approximation algorithms, the best solution may not necessarily found but a set of constraints have to be satisfied. Most of the physical design optimization problems are NP-Hard problems that can't be solved in a polynomial time and consequently is to be satisfied with approximate solutions [16].

Heuristics can be categorized into constructive heuristics and iterative heuristics. Constructive heuristics generate solutions according to some specific criterion. Once a reasonable solution has been found, these algorithms do not attempt to improve it. Iterative heuristics do not try to reach a feasible solution; rather they start from one of them (obtained perhaps through a constructive method) and then improve it through an iterative process that ends when a given constraint is fulfilled.

Evolutionary optimization is one of the most important concepts that have been heavily exploited in the iterative heuristics. This concept is based on evolutions that exist in the nature and it is try to imitate this natural process to solve hard problems. Every evolutionary algorithm works on a set of individuals called a population and it should well-define two important steps: selection and mutation. In the selection step a subset of the population is selected based on some criteria, and then this subset will be processed in the mutation step trying to enhance its quality [17].

Simulated Evolution (SimE) is an evolutionary optimization algorithm that has been used in a variety of applications and its keep proving that it can achieve good results when applied in suitable and well-defined way. [18] used this algorithm to solve the 2D standard cell placement problem and it produced good results with less processing time. The following subsection explained this algorithm step by step in details.

1.3.1 Simulated Evolution

SimE is one of the evolutionary iterative algorithms that can be applied and adapted for a variety of optimization problems in different areas. Along the years, SimE shows it can give good results and outperform some other iterative algorithms such as Simulated Annealing when applied to different problems. The structure of the SimE algorithm is shown in Figure 1.5 [16].

SimE assumes that there exists a solution $\boldsymbol{\varphi}$ of set **M** of **N** (movable) modules. The algorithm starts from an initial assignment $\boldsymbol{\varphi}$ initial, and then, following an evolution-based approach, it seeks to reach better assignments from one generation to the next by perturbing some ill-suited components and retaining the near-optimal ones [16].

A cost function Cost associates with each assignment of movable element $\mathbf{m_i}$ a cost $\mathbf{C_i}$. The cost $\mathbf{C_i}$ is used to compute the goodness $\mathbf{g_i}$ of an element $\mathbf{m_i}$, for each $\mathbf{m_i} \in \mathbf{M}$. The algorithm has one main loop consisting of three basic steps, Evaluation, Selection, and Allocation. The three steps are executed in sequence until the solution average goodness reaches a maximum value, or no noticeable improvement to the solution fitness is observed after a number of iterations. The Evaluation step consists of evaluating the goodness $\mathbf{g_i}$ of each element $\mathbf{m_i}$ of the solution $\boldsymbol{\varphi}$.

ALGORITHM *Simulated_Evolution*$(B, \Phi_{initial}, StoppingCondition)$
NOTATION
B= Bias Value. Φ= Complete solution.
m_i= Module i. g_i= Goodness of m_i.
$ALLOCATE(m_i, \Phi_i)$=Function to allocate m_i in partial solution Φ_i
Begin
Repeat
EVALUATION:
ForEach $m_i \in \Phi$ evaluate g_i;
/* Only elements that were affected by moves of previous */
/* iteration get their goodnesses recalculated*/
SELECTION:
ForEach $m_i \in \Phi$ **DO**
begin
IF $Random > min(g_i, 1)$
THEN
begin
$S = S \cup m_i$; Remove m_i from Φ
end
end
Sort the elements of S
ALLOCATION:
ForEach $m_i \in S$ **DO**
begin
$ALLOCATE(m_i, \Phi_i)$
end
Until *Stopping Condition is satisfied*
Return Best solution.
End (*Simulated_Evolution*)

Figure 1.5: General Structure of SimE Algorithm [16].

The goodness measure must be a single number expressible in the range [0, 1]. It is defined as:

$$g_i = \frac{O_i}{C_i} \quad \text{..... (eq. 1)}$$

Where $\mathbf{O_i}$ is an estimate of the optimal cost of element $\mathbf{m_i}$, and $\mathbf{C_i}$ is the actual cost of $\mathbf{m_i}$ in its current location. The second step of the SimE algorithm is Selection. Selection takes as input a bias value **B**, the solution $\boldsymbol{\varphi}$ together with the estimated goodness of each element. It partitions $\boldsymbol{\varphi}$ into two disjoint sets; a selection set **S** and a partial solution $\boldsymbol{\varphi_p}$ of the remaining elements of the solution $\boldsymbol{\varphi}$. Each element in the solution is considered separately from all other elements. The decision whether to assign an element **mi** to the set **S** is based solely on its goodness $\mathbf{g_i}$. The selection operator has a non-deterministic nature, i.e., an individual with a high goodness (close to one) still has a non-zero probability of being assigned to the selection set **S**. It is this element of non-determinism that gives SimE the capability of escaping local minima.

Allocation is the SimE operator that has the most important impact on the quality of solution. Allocation takes as input the set **S** and the partial solution $\boldsymbol{\varphi_p}$ and generates a new complete solution $\boldsymbol{\varphi_N}$ with the elements of set **S** mutated according to an allocation function. The goal of Allocation is to favor improvements over the previous generation, without being too greedy [16].

1.4 Force Directed Algorithm

Computational hard problems sometimes solved using numerical techniques that can be easily adapted to solve the desired problem. To utilize these techniques, the problem has to be transformed into a numerical problem. Numerical solutions usually involved some mathematical computations that can be executed in a deterministic way. Like any other hard problem, the placement problem could be transformed into a numerical optimization one that consequently could be solved using any of the proper numerical approaches.

One of the well-known numerical techniques is the force directed (FD) method; this method basically inspired by the Hook's law in mechanics which stated that the force exerted on a spring to extend or compress it by some distance is proportional to that distance. Also if a free body is connected to other bodies by springs, then it will stop at a zero-force location where the resultant force exerted on it is minimal. [19] utilize this concept to solve the placement problem after transform it to a numerical optimization

problem. This method can be summarized as follows: consider a module **i** that is connected to different modules in the same circuit. Let $\mathbf{d_{ij}}$ denotes the distance between the two modules **i**, **j** and $\mathbf{w_{ij}}$ is the weight of this connection; then the total force exerted on module **i** can be computed using (eq.2).

$$F_i = \sum_j w_{ij} * d_{ij} \quad \ldots\ldots\ldots\ldots \text{(eq. 2)}$$

In 2D placement problem each module is affected by forces in both x and y directions. To place this module at a good location, we can place it at its zero-force location by equalizing all the forces that applied on it to zero. Let X_0 and Y_0 denote the coordinates of the zero-force location in x and y directions respectively, to find this exact location we have to solve the following two equations:

$$\sum_j w_{ij} *(X_0 - x_j) = 0 \quad \ldots\ldots\ldots\ldots \text{(eq. 3)}$$

$$\sum_j w_{ij} *(Y_0 - y_j) = 0 \quad \ldots\ldots\ldots\ldots \text{(eq. 4)}$$

Where x_j, y_j are the coordinates of module **j** current location. Solving (eq. 3) and (eq. 4) For X_0 and Y_0,

$$X_0 = \frac{\sum_j w_{ij} * x_j}{\sum_j w_{ij}} \ldots\ldots\ldots\ldots \text{(eq. 5)}$$

$$Y_0 = \frac{\sum_j w_{ij} * y_j}{\sum_j w_{ij}} \ldots\ldots\ldots\ldots \text{(eq. 6)}$$

Another important issue in this approach is to resolve the situation where more than one module has the same zero-force location. This can be solved by different ways, for example placing the module to the nearest free location.

In Our work we extend this algorithm to solve the 3D placement problem. In 3D IC there is another direction (z-direction) where a force can be applied to the module; so to extend FD to cover this case all what we need to do is to solve a third equation (eq. 7) that will generate the best layer (Z_0) for placing the desired module.

$$Z_0 = \frac{\sum_j w_{ij} * z_j}{\sum_j w_{ij}} \ldots\ldots\ldots\ldots \text{(eq. 7)}$$

1.5 Multi-Objective Optimization

In many optimization problems the designers are targeting to optimize multiple objectives in the same problem. Most of the times these objectives are conflicting objectives which accordingly introduce a tradeoff in the optimization process. To resolve this issue the designer need an efficient way that gives him the ability to combine conflicting objectives and work around this tradeoff in a proper manner.

One of the most effective approaches to handle the Multi-objective optimization is the fuzzy logic. Unlike ordinary binary (Boolean) logic where things are either true or false; fuzzy logic is a many valued logic where things can be partially true or partially false. This logic has been introduced by lotfi zadeh in 1965 [20]. In binary logic, variables may have a value that is either 0 or 1. On the contrary, fuzzy logic variables may have a truth value that ranges from 0 to 1; this truth value called the degree of membership of that variable. Since some times the variables may have values outside the range [0, 1]; a membership function can be defined to map these values into a values between [0, 1].

In multi-objective optimization problems the different objectives may have different dimensions and different ranges; so to solve this, the designer can define a membership function for each objective and this will produce normalized dimensionless objectives that are ready to be used in the combining step. Defining a representative membership function requires a deep understanding of the problem domain and sometimes knowledge about the extreme values that the variable may take. To clarify the concept, consider the following example; let **v** refer to the number of vehicles in the parking, and assume we need to define a membership function that will generate an indication of the parking congestion. To be able to define a suitable membership function we need to know the maximum and minimum number of vehicles that can exist in the parking at the same time. The congestion is 0 when the parking is empty (0 cars) and 1 when the parking is full (MAX cars); knowing this information we can define the

member function as a linear function that connect those points (0, 1) and (MAX, 1). Figure 1.6 shows the graph of this function.

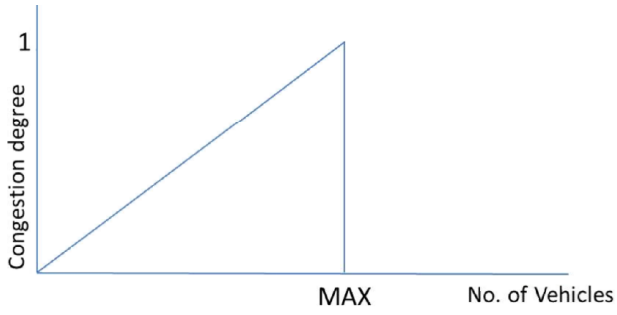

Figure 1.6: Example of a Membership Function

To combine different objectives after normalizing them using well defined membership functions; many rules and methods are exist. The simplest and one of the elegant and effective ways is the weighted sum. In this method a specific weight between (0, 1) is assigned to each objective and the weighted sum of the membership values is considered as the combined objective value between (0, 1). For example if we have to combine two objectives T and W to a single combined objective C; we need first to normalize these objectives and then assign a weight for each of them. Let B denotes the weight for objective T, then the combined objective is:

C = B * Normalized (T) + (1-B)* Normalized (W) ……. (eq. 8)

The issue with the weighted sum is that the designer should assign the weights carefully and sometimes an experiment is needed to be conducted to find the suitable weights empirically.

1.6 Motivation and Objectives

3D IC technology is promising technology that overcome a lot of 2D technology limitations such as long interconnect delay and large flat IC area. The main problem with this technology is that its physical design problems are complicated ones due to its large solution space. Also, since TSV plays a dominant role in this technology, its limitations have to be considered carefully and efficiently. The main TSV limitation is its large size

compared to the modules itself and though it is consumes a significant amount of silicon area. Also if these TSVs are not planned well they will considerably degrade the solution quality and the routing possibilities.

Consequently, there are abundant researches, methods and approaches that handle the different issues in the 3D IC physical design. But looking to these approaches only few of them are using the iterative heuristics to tackle these problems, and actually their focus is only on the SA and FD approaches. Since the iterative approaches especially the evolutionary heuristics have shown very good results when applied to a variety of problems specifically the 2D physical design problems, it's a traditional and a good practice to explore the effectiveness of this class of algorithms on this new technology.

1.6.1 Objectives

This work investigates the challenges that appear in the 3D IC physical design process, and especially the ones that are imposed by the TSV. The main objective of this research is to experiment and apply different non-deterministic approaches and target the minimization of the TSV count in multi-layer partitioning (Part 1), and also to reduce the silicon area of each partition by considering WL as a second objective (Part 2).

The following tasks will be carried in this work:

1- Defining a representative State Representation (Data Structure) that efficiently captures the structure of the 3D chips, and is suitable for efficient computation of cost, and its update due to incremental moves.
2- Formulating a typical Cost Function. This function will include the Wire length and the TSV count.
3- Identifying a suitable Goodness Measure for movable elements, this is a requirement prior to selecting and re-allocating movable elements which in our case are cells.
4- Engineering the SimE heuristic for the 3D placement problem.
5- Comparing the Simulated Evolution results with the state of art methods and other previous iterative approaches using published benchmarks.

1.7 Methodology

In 3D IC design the solution space is huge, and consequently the time required to explore it is tremendous. To traverse the search space efficiently, as a first step we need to come up with an efficient data structure that can be used to represent the state/solution. Both the space requirements to hold the solution and the time needed to determine the cost of the solution are critical. To achieve that we need to study the advantages and disadvantages of possible proposed representations and identify the best one.

For the objective function in the 3D placement problem there is a significant tradeoff between the different parameters that define it. So we need to formulate the cost function by identifying a reasonable weight for each component. The weight will be directly related to the component's role and importance in the problem. Use of fuzzy logic to combine multi-objectives will be investigated. Also we have to develop the cost function to be incrementally updated in order to make the iterative evaluation based on move more efficient.

Simulated Evolution heuristic can be employed to explore search space of the 3D placement problem. Unfit elements are selected and reallocated until the solution is evolved. The goodness function should reflect the desired characteristics of the problem and be tightly coupled with the objective function being optimized. In order to validate our work we need to continuously observe the overall individuals goodness and make sure that it keeps increasing, and the size of the selection set that comprises elements to be continuously decreasing. Force-directed placement heuristic will be experimented in allocation step of SimE.

1.8 Summary

This Chapter has introduced the basic background and information that is required to understand the 3D VLSI technology, the physical design phase, and the iterative algorithms that can be employed to optimize most of the hard problems in the physical design stage. Additionally, in this Chapter the significance and importance of our work

has been stated, also the main objectives and the followed methodology have been explained.

The rest of this thesis is organized as follows. Chapter 2 explores the previous work in the literature that is related to our work and differentiates our approaches from the others. Chapter 3 is the first part of our work which is the TSV minimization in 3D IC designs under the area balancing constraint; in this Chapter the problem formulation is specified first, then the adopted solution is explained, and finally the results and comparison with previous work is stated. Chapter 4 discusses the second part of our work which is about the placement problem in 3D cells design; the problem statement and the developed approach are stated first, after that the results are summarized and discussed. The thesis is concluded with a conclusion Chapter that summarized the main problem under study, the developed solutions, the outcome and contribution of our work, and finally the recommendations and future work that can be explored further more.

CHAPTER 2

LITERATURE REVIEW

Like 2D VLSI design, 3D VLSI Design attracted the attention of researchers because of its importance and criticality. The physical design aspects of 3D VLSI along with TSV technology have been explored deeply in the literature in many different ways; before we proceed to the detailed exploration of the previous work, it's helpful to classify the main issues that were studied in the literature and summarize the most important adopted approaches. These issues are basically related to either the 3D physical design in general or to the TSV technology. Floorplanning, module placement and routing are the main physical design issues that have been discussed thoroughly. For TSV, the main considered issues are: TSV minimization, TSV insertion and Co-placement, TSV assignment, and other thermal issues.

Floorplanning in 3D VLSI design is the most important aspect that has been covered in the literature, and it's handled using a variety of approaches that either extend the 2D approaches or develop new methods that rely mainly on simulated annealing (SA), generalized slicing tree or partitioning. The placement problem has been optimized analytically in some works; in other works a global placement has been developed followed by other steps; also partitioning and iterative force directed heuristic have been applied to this problem. A lot of work also was done on the optimizing of routing in 3D VLSI design.

Regarding TSV minimization, integer linear programming (ILP), dynamic programming, modified Fiduccia-Matthyses (FM) based heuristics and iterative partitioning and compaction techniques have been exploited. The process of TSV minimization has been considered as a pre-independent step of processing in some works while some other researches mixed it with placement and floorplanning. In co-placement approaches the TSV and modules have been considered simultaneously using different constructive and iterative approaches.

In TSV assignment the main goal is to assign preplaced TSVs to the different nets, and this process has been carried by min-cut max-flow algorithms, binary ILP, minimum spanning tree, and shortest path search. Thermal issues and power dissipation also was discussed thoroughly in the literature because of the heat and temperature considerable effects. Now we are going to explain and highlight the main issues that researchers addressed in their work along with the main developed methods.

2.1 TSV Minimization

One of the basic and early issues in the 3D IC design is the TSV minimization. In this process the objective is to minimize the number of signal TSVs that interconnect 3D nets. Zhong, et al., Cong, et al, Hsu, et al., and Huang, et al. [21-25] assumed that each 3D net will utilize only one exclusive TSV between any two adjacent layers. Pathak, et al. [26] discussed that this assumption may degrade the overall quality because using few TSVs may increase the WL while utilizing too many will enlarge the total silicon area. This means that TSV minimization should be only an early stage in the 3D design flow. In our work we follow the common assumption in the literature where each 3D net will use one TSV between adjacent tiers.

Goplen and Sapatnekar, and Cong and Luo [27-29] mixed the TSV minimization with other design stages such as floorplanning and placement. On the other hand, Li, et al., Pavlidis and Friedman, and Huang, et al. [30, 31, 22] considered this stage as an independent and preprocessing stage. Li, et al. [5] handled the TSV minimization by first placing the modules in 1D placement, and then they used a dynamic programming approach to determine the best cut positions under the constraint of area balancing. In our work we handle the TSV minimization along with the area balancing together in the first part; also we optimize both the WL and TSV at the same stage in the second part.

The partitioning approach has been deployed to achieve TSV minimization; the problem with the traditional multi-way partitioning is that the minimized cut size can't reflect how well the usage of TSVs is optimized. Moreover, the arrangement of these partitions could be non-trivial. So in most partitioning based approaches, the multi-way partitioning has been modified to accommodate 3D IC designs.

Pathak, et al. [26] developed an iterative partitioning approach that includes cuts in the z-direction, and they showed that the order of z-cuts is critical and important. A modified FM based partitioning approach was applied to resolve the TSV minimization problem by Hu, et al., and Kim, et al. [32, 33]; furthermore, Hu, et al. [32] considered the area constraint for each layer. Our work incorporates the FM approach as one of the goodness measures of SimE, and also we consider the area balancing as an important constraint when minimizing the TSVs.

Goplen and Spatnekar [28] considered the TSV minimization as a part of the global placement stage and they handled it using recursive bisection method. Jiang [34] applied an ILP approach to optimize the number of TSVs. Most of the mentioned works consider the number of TSVs without any consideration of TSV distribution. Huang, et al. [22] minimized the TSV count while smoothing the distribution of them in 3D structures using an iterative partitioning layer aware approach. They utilized the hmetis engine for the partitioning process and they introduced the concept of compaction and supervertix. Also they considered the IO pad constraint. Sait, et al. [35] depicted the TSV minimization problem as a layer partitioning problem under the area balancing constraint; they resolved it by applying the SA and Tabu Search (TS) algorithms; they also considered the IO pad constraint; since their assumptions and constraints are similar to our work, we compared most of our results in the first part to their outcomes.

In the first part of this thesis we optimize the number of TSVs without any consideration of their locations; but in the second part we optimize the TSVs count while placing them concurrently with the modules with the target of optimizing the overall WL. Also we consider the location of IO pads as a constraint in all of our work.

2.2 Floorplanning and Placement

The other important issues in 3D IC physical design are floorplanning and placement. To make things easy, we can classify the 3D floorplanning and placement approaches to two main classes: Thermal-aware, and Thermal-unaware. First we will discuss the thermal-unaware approaches which mainly target the minimization of WL, total area, and TSV count.

Simulated annealing is one of the main approaches that were applied for optimizing the placement and floorplanning stages. Tsai, et al. [8] developed a method that places the TSV and modules together considering the size of TSVs. The method consists of two stages: the annealing stage and the refinement stage. The results showed a high successful rate of their method. He, et al. [36] integrated the buffer insertion (to improve interconnect delay) and TSV planning together in the floorplanning stage. Their method applies the SA at three phases: area optimization, timing optimization, and TSV and buffer planning, the method has a linear complexity. They also considered the white space redistribution (WSR) for TSV insertion; they applied a heuristic that utilize the constraint graphs. The problem with this method is that white space within the restricted feasible region may not always be sufficient to accommodate buffers and TSVs and consequently it may not guarantee completed routing. Zhong, et al. [21] used the minimum spanning tree (MSB) for WSR.

Hierarchical floorplanning was studied and exploited in some previous works. Li, et al. [30] approach scales the 3D floorplanning problem down to reduce solution space. They handled it by considering a several 2D floorplans together; they neglected the TSV size to lower down the design complexity. Adya and Markov [37] introduced a new concept of making ‘moves’; the moves in their work are based on something called floorplan slack.

Non-SA based approaches have been incorporated a lot in the floorplanning optimization. Li, et al. [5] developed a constructive approach that utilizes the hypergraph partitioning and the generalized slicing tree for floorplanning. Chen and Yoshimura [38] defined a new floorplan representation that decreases the number of floorplan configurations, the TSV size is not considered. Lu, et al. [39] focused on finding the locations for all the TSVs such that the estimated chip performance is maximized. This method depends on the given floorplan. Since in this thesis we are considering the cells with comparable sizes, the problem of floorplanning optimization is not applicable; therefore we didn’t consider it in our work.

Another group of researchers handle the placement problem analytically. Hsu, et al. [23, 24] developed a three steps fast analytical method that considers the TSV size for

3D placement; analytical global placement takes place first (with layer assignment), then a legalization 2D technology based step is applied (considering the connection between layers) along with TSV insertion, finally layer-by-layer detailed placement is carried out. Cong and Luo [29] applied a multilevel analytical 3D placement that utilizes a nonlinear optimization method to handle the global placement. After global placement an overlap removal and device layer assignment process is applied by adding a density penalty function for both area and TSV density constraints. The method neglects the TSV size.

Kim, et al. [40, 33] considered the placement of TSVs before the modules, and the co-placement of them together. Their approach is a modified force directed method that perform placement in each die separately. Our work focus on the placement problem in 3D cell designs; the approach we followed is an iterative (non-deterministic non-constructive non-analytical) approach that places the modules and the TSVs simultaneously and utilizes the force directed method as one of its important step.

Thermal-aware approaches consider the heat and temperature along with the area and the wirelength. So there is a new kind of silicon via that normally used in this class of approaches, this new kind is called the Thermal Through silicon via (TTS) and it is only used as thermal conduits and has no electrical function [41]. Goplen and Spatnekar [28] addressed the thermal issue analytically; they developed an analytical partitioning based approach with terminal propagation. They used a net weighting to reduce the length of nets with high-power usage and high thermal resistance. Thermal resistance reduction nets are created to move cells toward areas of lower thermal resistance based on their power dissipation. The method ignores the TSV size but it has a linear complexity. Li, et al. [42] provided a novel analytical algorithm to re-allocate white space for 3D ICs to facilitate via insertion thermal resistive model. They used a LP based thermal aware approach.

Goplen and Sapatnekar [27] applied an iterative force-directed approach in which thermal forces direct cells away from areas of high temperature. Their work is based on finite element analysis; the results showed lower maximum and average temperatures with wirelength increasing slightly. Cong, et al. [43] developed a three-stage force-directed to optimize peak temperature, area, wire length, and via count.

Cong and Zhang, and Li, et al. [44, 45] formulated the vertical thermal via distribution as convex programming problem. Li, et al. [45] used the SA to generate floorplans of all layers simultaneously after performing the vertical via distribution. Cong and Zhang [44] formulated a thermal through silicon via minimization problem with temperature constraints as a constrained nonlinear programming problem (NLP) based on the thermal resistive model. Also the horizontal via planning is based on two efficient techniques: path counting and heat propagation based on the heat dissipating path analysis.

Cong, et al. [25] considered the thermal effect and area impact of TSV on a preplaced cells under area density constraint. They utilized the signal TSV only. Cong, et al. [46] transformed a 2D placement into a 3D placement. They defined an effective thermal cost but they ignored the TSV size. Li [47] addressed the thermal via planning in a preplaced 3D IC by an improved thermal aware gravity algorithm. Zhang, et al. [41] developed a temperature-aware 3D global routing algorithm; the algorithm inserts thermal vias and thermal wires to reduce chip temperature. The main phases of the work are: routing congestion estimation, TSV assignment, thermally driven maze routing, and LP based thermal via/wire insertion.

Wong and Lim [48] introduced the first thermal analyzer based on random walk techniques. They developed several ways to achieve their goals. The first way is an area/wirelength driven floorplanning based on SA to optimize the area and wirelength. The second way is thermal driven floorplanning based on SA to optimize the area, wirelength and temperature. The last way is an integrated thermal via floorplanning thermal via insertion algorithm where each time during SA it take the effect of thermal vias into consideration. Thermal issues are out of this thesis scope so they haven't been considered in our work.

Finally in regards to the outline constraint, Adya and Markov, Xiao, et al., and Tsai, et al. [37, 49, 8] satisfied the fixed outline constraints and addressed the soft modules in their work while Chen and Yoshimura, and Li, et al. [38, 5] considered it as a penalty in the cost function. In our work the outline is defined by the number of rows in each layer and the number of cells in each row; the outline is same for all layers, we

selected a suitable fixed outline that can handle all the modules as well as minimizing the empty left cells. Hard modules only have been considered in this thesis.

2.3 TSV Assignment

The last important stage in the 3D IC physical design is the TSV assignment. As mentioned earlier, the main goal of this stage is to assign preplaced TSVs to the different nets. Tsai, et al., and Li, et al. [8, 5] carried out this stage using a Minimum Cost Maximum Flow (MCMF) method. Kim, et al. [40, 33] solved this problem using MST; the 3D nets (a net with terminals on multiple tiers) are sorted in ascending order according to their bounding box and then the algorithm selects the nearest TSV to the shortest edge. Yan, et al. [50] applied several methods to perform the assignment; these methods are: binary ILP, the Hungarian method, an approximation algorithm that is accurate and fast, and a local search based heuristic (neighborhood search method) which is the fastest but less accurate.

Liu, et al. [51, 52] adopted the following approaches: shortest path search, bipartite matching, and (MCMF). To achieve high quality solution, first they found the shortest path of each 3D net and got the lower bound of the total wirelength; this method may produce illegal shared assignment. Then weighted bipartite matching and (MCMF) are applied to split nets that share the same TSVs and a feasible assignment solution is derived. Liu, et al. [52] applied an ILP to further optimize the TSV assignment. Liu, et al. [51] overcame the limitation of Liu, et al. [52] work which is considering only 2-pin nets, by deploying a decomposition method. In our work the TSV assignment is handled by assigning the TSVs to the nets first (randomly); this will fix each TSV to a specific net and a specific layer, then this assignment is considered in the placement process automatically.

2.4 State Representation and Cost Formulation

Another important issue in the 3D physical design is the state representation. Mainly in the literature, for the partitioning problem, the hypergraph is the most commonly used representation [23, 22, 24, 5, 35] . For floorplan representation there are

a number of adopted representations, the most important ones are: array of sequence pairs [37, 49, 8], partitioned sequence pairs (PSP) [38], corner block list (CBL) [36], combined bucket and 2D array (CBA) [43], and slicing trees [53, 5]. Sequence pair consists of two permutations orderings of the modules. The two orderings represent topological relations between modules in each layer. In sequence pair every two blocks constrain each other in either vertical or horizontal direction. Multiple sequence pairs may encode the same module placement. PSP Represent multi-layer floorplans efficiently; the number of configuration of 3D IC floorplans represented by PSP is less than that of planar floorplans represented by sequence pair and decreases as the device layer number increases. One sequence pair can generate multiple PSPs. CBL represent the topological relations between modules of each layer. The whole layout of one layer can be partitioned into rooms, and modules are packed within rooms.

CBL is an efficient topological representation for non-slicing floorplan and it is independent of the module sizes. It requires the same computing complexity of binary trees and less than sequence pair. It can represent all floorplans with slicing structure, and represent non-slicing floorplans [54]. CBA exploit the 3D solution space efficiently; in CBA the physical constraint in z- axis is different from x-axis and y-axis, no packing is necessary in z-direction, and the position relationship can be derived directly from the layer number of each block. A 2D floorplan representation is used to represent each layer, and a bucket structure to store the relationship between blocks at different device layers. In the first part of our work we used the hypergraph representation because it is suitable and simple for the TSV minimization which is kind of a partitioning problem. For the second part we represented the 3D chip as a multilayer stack where each layer is 2D grid of rows and columns, the intersection of a row and a column is a cell that can handle exactly one module. The grids of all layers have the same predefined size.

Before we conclude this Chapter, it is important to summarize the main 3D IC physical design cost functions in the literature. The most commonly considered components of the cost function is the area, WL, the TSV count, the aspect ratio (AR), the fixed outline penalty, and the thermal cost (temperature). Which components to include is a problem specific and objective dependent decision. The area mainly includes

the modules area and sometimes the TSV area. For the TSV count, Huang, et al. [22] calculated it by summing all the spanned layers of all nets. The thermal cost can be expressed as the deviation from a thermal optimal solution [25].

For the wirelength there are a number of estimations that is summarized as follows: Some researchers estimated the WL of a 3D net by the semi perimeter (SP) of the bounding box of all its terminals as if they were in the same layer [43, 50, 30, 46, 55, 38, 49, 25], as shown in Figure 2.1(a). Hsu, et al. [23, 24] used a weighted average version of this estimation and Li, et al. [30] developed a statistical one that is automatically balanced the area. In the mentioned estimations the TSV location is not considered at all. Tsai, et al. [8] considered the bounding box of all terminals and TSV associated with 3D net as in Figure 2.1(b). This estimation provides more realistic values but still underestimates the WL for a net with terminals in multiple tiers. Li, et al. [5] estimated the total WL of a net by summing up its WL on each individual layer that it spans. The WL of a net on a specific tier is calculated as the SP of the bounding box of the terminals of that net in that die and any TSV of the net in the same die or the die above. Figure 2.1(c) explains this method. In our work we consider the area of modules and the area of TSVs. The TSV count for each net is calculated as the number of layers that the net spans; the WL was estimated using the method in Figure 2.1(c).

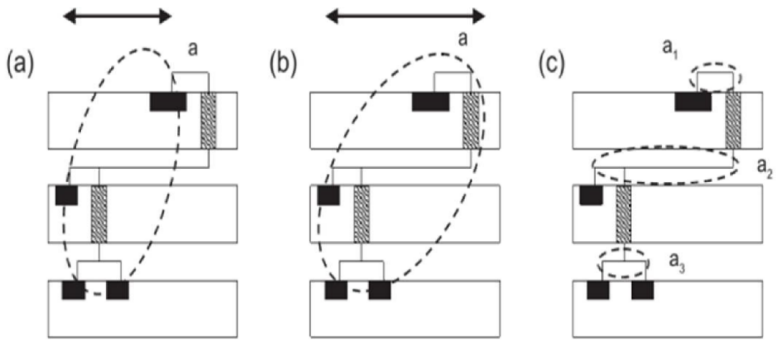

Figure 2.1: WL estimation models. (a) Bounding box of all terminals of a net. (b) Bounding box of all terminals and TSVs of a net. (c) Break into subnets and sum up their lengths [5].

2.5 Summary

In conclusion for this review, we can observe the importance of the 3D IC VLSI technology, and the wide area of issues that has been discussed and the variety of problems that need further studying; all these issues along with the different parameters and factors that have to be considered in the physical design of 3D circuits, make this technology a hot area for research and development. Also as it clear in the literature, the deployment of iterative algorithms is not extensive which make it worthy to apply some well-known iterative heuristics (SimE) on the optimization of 3D physical design problems and test its effectiveness. Another important thing that we can notice by looking to the previous researches is that no one has considered the 3D cell placement problem in its common and default context; therefore exploring such problem is important and necessary.

In the first part of this work we apply the SimE algorithm for TSV minimization considering the area and IO constraint; this work is explained in Chapter 3. The second part of this thesis is the optimization of both the WL and TSV count in the 3D cell designs; this work is illustrated in Chapter 4.

CHAPTER 3

TSV OPTIMIZATION UNDER AREA BALANCING CONSTRAINT

TSV is a suitable and promising technology that can achieve the interlayer connection in a very elegant and robust way. The main problem of this technology is the significant area of TSV compared to the functional modules area. This large difference in area along with other thermal effects and mechanical considerations, highlight the importance of optimizing the number of TSVs used in any 3D chip. This optimization problem should take into account some constraints that are related the 3D design requirements; one of these important constraints is the balancing of the total area of all layers in the 3D chip.

In this part of our work we specified and applied different heuristics to minimize the required number of TSVs while at the same time satisfy the specified constraint. The developed solutions have been tested against a group of benchmarks that has different sizes (in terms of modules and nets) and different modules area standard deviation. The rest of this Chapter is organized as follow. The first Section explains the problem under investigation along with state representation adopted and the formulated cost function. After that, the developed solutions and the detailed steps of the applied algorithms are explored in the second Section. Finally, in the third Section the results are summarized and discussed and a comparison with some work in the literature is stated.

3.1 Problem Formulation

This part of our work investigates the TSV optimization problem that considers the area balancing as both a secondary objective and a constraint. The objective of this work is to minimize the required TSV Count (number of TSVs), and minimize the

standard deviation (SD) of the layers total area. Also in this problem we constrain the resultant solution to have an area coefficient of variance (COV) lower than 10%.

The TSV minimization problem is defined as follows:

Given the following inputs:

1) A set of design layers (numbered from 1 to L)
2) A set (V) of M modules and their areas
3) The set of IO pins (IOC pins)
4) The netlist (a list of all nets in the circuit (N nets))

We have to determine the layer where the module should be assigned such that the following objectives are minimized:

1) TSV Count
2) SD of layers area

And the following constraint is satisfied: the layers area COV should be less than 10%.

In solving this problem we assume the following:

1) A F2B stacking style is used
2) A via first TSV is considered
3) The IO pad is located at the bottom below all layers (at layer 0), this layer has zero area
4) Each net utilizes one exclusive TSV between two consecutive layers
5) To connect the modules of a single net in two consecutive layers, the TSV will be assigned to the top layer; e.g., if net *x* has modules in layer 1 and layer 2, then a TSV will be assigned to layer 2
6) All TSVs have the same area

3.1.1 State Representation

In this Section we explain how we represented the 3D chip and the netlist. The 3D chip consists of a number of layers that is stacked above the IO pad. Since in this part our concern is only assigning the modules to the corresponding layers, without and consideration of the module location within the layer, the 3D chip is represented as an **ordered** list of layers numbered from 0-L where layer 0 represent the IO pad and can contain only IO pins. The layers internal structure is not considered in this part, and each layer will keep information about the modules assigned to it, and the total area of each layer. Figure 3.1 explains this representation.

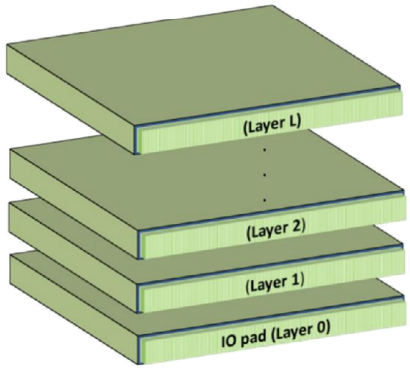

Figure 3.1: Representation of 3D Chip.

The netlist is presented as an array of linked lists, where each list is corresponding to one net and the array index corresponds to the net number. This actually is one way to present the hypergraph. In hypergraph, when more than two nodes are forming a net, the net is represented as a hyper-edge [56]. Each net is a hyper-edge (even the two nodes net can be considered as a hyper-edge) that is presented in our work as a linked list. To clarify things, consider the graph in Figure 3.2.

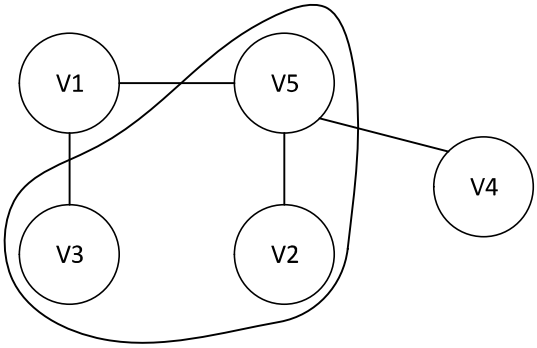

Figure 3.2: Example of a Hypergraph

The 2-nodes nets in the graph are represented as a direct edge; the 3-nodes net (v2, v3, v5) is represented as a hyper-edge. All these edges are represented in our solution by linked lists as shown in Figure 3.3.

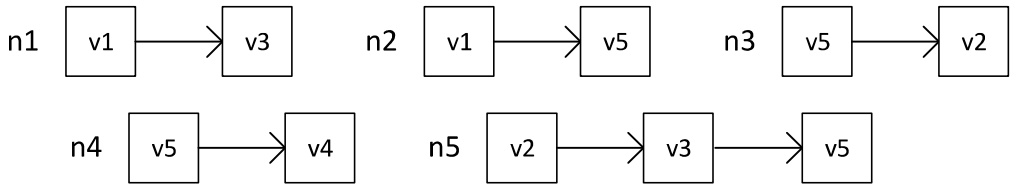

Figure 3.3: Representation of a Hypergraph as Netlist.

The connections between modules are captured as a 2D square matrix. The dimension of the matrix is the number of modules. The rows and columns are indexed by the module number. If there is a connection between module (i) and module (j), then the entry at the intersection of row (i) and column (j) [C_{ij}] will has a value of 1 otherwise its value is 0. In other words, row_i is a vector that depicts the connection information of module (i) with all other modules in the circuit. Consider an example where a circuit has three modules (v1, v2, v3), and v1 is connected to v2, and v2 connected to v3; the connectivity matrix will be as follow:

	V1	V2	V3
V1	0	1	0
V2	1	0	1
V3	0	1	0

3.1.2 Cost Formulation

In this multi-objective problem, there are two components contribute to the total cost: the TSV Count and the area SD. For TSV Count, each net uses a number of exclusive TSVs depending on the current assignment of its nodes. The TSV Count is the sum of all the required TSVs by all nets as formulated in eq. 9.

$$\text{TSV Count} = \sum_j TSVC_j \quad \text{(eq. 9)}$$

In eq. 9, $TSVC_j$ is the number of TSVs required by net j; $TSVC_j$ can be calculated as the number of layers spanned by the net. In other words, it is can be computed as the difference between the maximum and minimum layers that have a net's module. For example consider the following net; n (I2, V1, V4) where I2 is an IO pin in the layer 0 (IO pad) and V1 is located in layer 2 while V4 is assigned to layer 4. The value of $TSVC_n$ = max_layer – min_layer = 4-0 = 4.

The area SD represents the standard deviation of the layers' area. Assume AVG is the average of the areas of the L layers and A_i is the area of layer (i), then we can compute the area SD using eq. 10.

$$\text{SD} = \frac{\sum_{i=1}^{L}(A_i - AVG)^2}{L} \quad \text{(eq. 10)}$$

Since the SD is an absolute number; it is better to use a percentage measure that relate the SD with the average value, this measure is known as COV and can be calculated using eq. 11. Keeping this COV value under 10% is a constraint.

$$\text{COV} = \frac{SD}{AVG} \quad \text{(eq. 11)}$$

To be able to combine the TSV count and the area SD, we need to deploy the fuzzy logic. The first step is to define a suitable membership function for each objective. Since the target is to minimize both components; the membership value should has an inverse relationship with the absolute component value in both of them. One of the suitable and simple functions that can be used to capture this relation is the linear

function with negative slope. To make the function more realistic we need to define the extreme values of each component and after that we formulate the complete membership function of each one.

For TSV Count, the maximum possible number of TSVs occurs when all the nets have one IO pin at least and one module at least in the most top layer. The minimum number of TSVs occurs when all the modules are in layer 1. eq. 12 and 13 summarize these values.

$$TSV\_MAX = N * L \quad \text{(eq. 12)}$$

$$TSV\_MIN = IOC \quad \text{(eq. 13)}$$

Regarding the area SD, the minimum SD value occurs when all layers has the same area and it will be zero; according to the analysis of [57], the maximum SD of a set of values is less than or equal to 60% of the difference between the maximum and minimum values. The worst case in our problem is when all the modules are in the first layer; in this case the minimum area is zero (for upper layers) and the maximum area is the area of the first layer which consists of the modules areas and the area of all TSVs. The number of TSVs in this case is the TSV_MIN = IOC. eq. 14 and 15 summarizes these values.

$$SD\_MIN = 0 \quad \text{(eq. 14)}$$

$$SD\_MAX = 0.6*(\sum_{i=1}^{M} MA_i + IOC * TA) \quad \text{(eq. 15)}$$

Where MA_i is the area of module i, and TA is the TSV area.

Now, we can define the membership functions of the TSV Count and area SD as shown in Figures 3.4 and 3.5.

The mentioned membership functions are combined according to eq. 8 to produce the total combined objective (TCO) as shown in eq. 16. Considering the total combined

objective, the problem under investigation can be viewed as an optimization problem with the target of maximization the total combined objective (TCO).

TCO = Bc * Normalized (TSV Count) + (1-Bc)* Normalized (SD)..... (eq. 16)

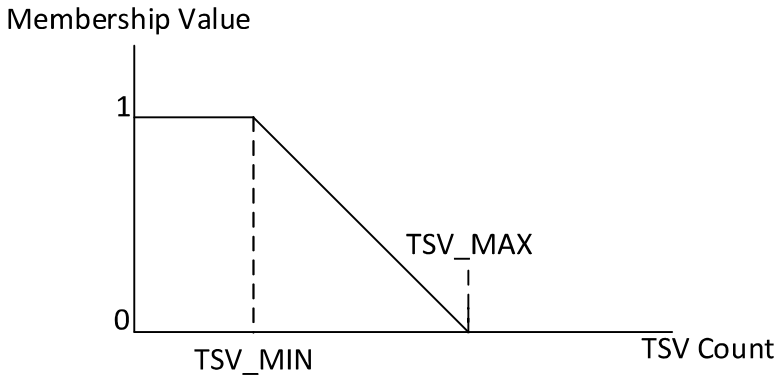

Figure 3.4: Membership Function of TSV Count.

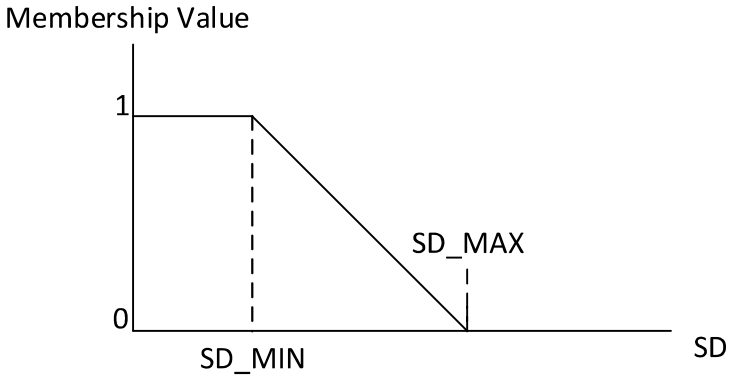

Figure 3.5: Membership Function of Area SD.

3.1.3 Benchmark Circuits

In order to test the quality of our solution, we have to test it on a collection of different well-known Benchmark Circuits. We selected a set of eight Benchmark Circuits that are differ in the number of nodes, the number of IO pins, the number of nets, and the modules area SD. These Benchmarks are Available at: http://vlsicad.cs.binghamton.edu/benchmarks.html. Table 3.1 summarizes the main characteristics of the used Benchmarks.

Table 3.1: Characteristics of Benchmark Circuits.

Name	No. of Modules	No. of IO pins	No. of Nets	Modules Area SD
n100	100	334	885	906
n200	200	564	1585	479
n300	300	569	1893	482
tseng	2417	174	2295	1
diffeq	3024	103	2985	1
elliptic	6831	245	6717	1
frisc	7024	136	6908	1
pdc	7609	56	7569	1

3.2 Developed Approach

The problem under investigation is a multi-objective hard problem that targets the minimization of both the TSV Count and the area SD. To solve this problem we adapted and customized a well-known efficient iterative heuristics. Two methods have been developed, applied, and tested on the problem: a goodness based SA algorithm (GBSA), and the SimE algorithm. The subsequent subsections explain in details the proposed methods.

3.2.1 Goodness Based SA (GBSA) Approach

Simulated Annealing (SA) is one of the famous and oldest heuristics that had been applied repeatedly to solve a lot of hard problems. Sait, et al. [35] applied the standard SA algorithm to solve the TSV minimization problem. Before we proceed with the algorithm we have to mention that the meaning of solution from now onward is a one assignment of the all modules to the corresponding layers. The general structure of the

algorithm is shown in Figure 3.6. Refer to Sait and Youssef [15] for more details about the SA algorithm and its application in physical design.

The SA algorithm started from an initial random solution, and then it continues for a predetermined number of iterations (Maxtime). The key parameter in SA is the annealing temperature (T), this parameter started at initial T_0 value and it is reduced in every iteration by a factor of (α). The role of this parameter is to control the random acceptance of bad solutions. The Metropolis function is called every iteration to alter/improve the current solution. Figure 3.7 highlights the details of the Metropolis function.

Algorithm Simulated_annealing($S_0, T_0, \alpha, \beta, M, Maxtime$);
(*S_0 is the initial solution *)
(*BestS is the best solution *)
(*T_0 is the initial temperature *)
(*α is the cooling rate *)
(*β a constant *)
(*$Maxtime$ is the total allowed time for the annealing process *)
(*M represents the time until the next parameter update *)

Begin
$T = T_0$;
$CurS$=S_0;
$BestS$=$CurS$;/* $BestS$ is the best solution seen so far */
$CurCost$=*Cost*($CurS$);
$BestCost$=*Cost*($BestS$);
$Time = 0$;
Repeat
Call Metropolis($CurS$, $CurCost$, $BestS$, $BestCost$, T, M);
$Time = Time + M$;
$T = \alpha T$;
$M = \beta M$
Until ($Time \geq MaxTime$);
Return ($BestS$)
End. (*of Simulated_annealing*)

Figure 3.6: General Structure of SA Algorithm [15].

Metropolis function alters the current solution by calling the Neighbor function. The Neighbor function selects two random modules from two different layers and swaps

them. If the produced solution has a lower cost than the last one, the swapping will be committed and the solution will be accepted. Otherwise, a random number between [0, 1] is generated and compared to the stated exponential formula; if the random number is less than the formula evaluation, the solution will be accepted otherwise it will be rejected. As one can observe in the exponential formula, when T is large (at early iterations) the probability of accepting bad solutions is high, but as the algorithm progresses and T decreases the probability decreases and the algorithm become more greedy (tends to accept only better solutions). This process is repeated M times in each call of the Metropolis function.

Algorithm Metropolis($CurS$, $CurCost$, $BestS$, $BestCost$, T, M);
Begin
  Repeat
    $NewS$= $Neighbor$($CurS$); /* Return a neighbor from $CurS$ */
    $NewCost$=*Cost*($NewS$);
    $\Delta Cost$=($NewCost$−$CurCost$);
    If ($\Delta Cost$< 0) **Then**
      $CurS$=$NewS$;
        If $NewCost$< $BestCost$ **Then**
          $BestS$= $NewS$
        EndIf
    Else
      If ($RANDOM < e^{-\Delta Cost/T}$) **Then**
        $CurS$=$NewS$;
      EndIf
    EndIf
    $M = M - 1$
  Until ($M = 0$)
End. (**of Metropolis**)

Figure 3.7: Metropolis Function [15].

As we can notice from the above explanation, SA has many sources of randomness, and this may affect both the solution quality and the convergent time (time required for the algorithm to reach almost steady state solutions). To reduce this randomness and improve the solution quality while speeding up the algorithm, we developed the GBSA.

GBSA introduces only one major modification in the standard SA, specifically in the Neighbor function. Instead of selection the nodes on a random basis, GBSA proposes a goodness based selection. The goodness of a module is a value between [0, 1] that measures and reflects how good is the location of the module in the current solution. Higher goodness values indicate that the module is fit in its current location; on the other hand, lower values designate the badness of the current location.

In GBSA the Neighbor function consists of the following steps:

1. Goodness Evaluation
2. Selection of a subset from the modules set based on goodness values
3. Swapping two nodes from the selected subset produced in 2
4. Returning the altered solution

The goodness evaluation step is the first introduced step into the function. In this step the goodness of all modules is computed using GM_TSV_1 defined in Figure 3.8. GM_TSV_1 evaluates the goodness of a module based on two values, the current count of TSVs of all its nets (Actual), and the maximum number of the TSVs that all its nets can have (Limit). The lower the actual value is the higher the goodness will be and vice versa. In the second step a subset (PS) of the modules will be selected according to the SELECTION operation defined in Figure 3.9.

In the SELECTION operation, the modules are selected based on their goodness values; the higher the goodness is the lower the probability the module will be chosen. The BIAS value is a small value (less than 0.2) that can be either positive or negative; this value is usually used to adjust the selection inequality.

The remaining steps are the same as in SA, except now instead of selecting the two modules on a complete random basis, the two modules will be selected from the selection set (PS) only. This modification decreases the probability of swapping modules that are in a good location, and concentrates the swapping most of the time around the bad located modules.

Define GM_TSV_1

Input: module I

Output: Goodness [I]

Let *S* denote the set of all nets that has I as an element

Initialize *Actual*=0, *N* = Cardinality(S)

Limit = N * IOC

For all n in S

Actual = *Actual* +$TSVC_n$

End For

Goodness [I] = *1-(Actual / Limit)*

End Definition

Figure 3.8: Description of GM_TSV_1 Measure.

Define SELECTION

Input: Set *S* of *M* modules

Output: *PS* (Set of selected modules)

For all *I* in *S*

Generate *Random* in [0, 1]

If (*Random* < *Goodness* [*I*] + BIAS)

Append *I* to *PS*

End If

End For

End Definition

Figure 3.9: Description of the SELECTION Operation

3.2.2 SimE Approach

In this approach we customized the SimE algorithm explained in Chapter 1 to fit the problem under investigation. This subsection illustrates the main steps of the algorithm and how we defined each one. The general flow of this approach is explained in Figure 3.11.

After initializing the required variables and data structures and reading the Benchmark into the desired data structures, the initial solution is generated by assigning

the modules to the layers on a random basis; then the cost (TSV Count, area SD) of this solution is calculated along with the values of TSV_MAX, TSV_MIN, and SD_MAX. The remaining steps will be executed in a repeated manner until the stopping criteria is met; the stopping criteria is either reaching the predefined maximum number of iterations or when the overall combined cost remains almost constant for a specific number of iterations.

In the selection step, the SELECTION operation defined in GBSA is used to generate the selected set PS. Because later on, an allocation step will take place, all the selected modules will be removed from their locations and all the cost functions will be updated and calculated again to exclude the selected modules from consideration. The selected set PS size is constrained to contain at most 40% of the all modules.

In the sorting step, the modules in PS is sorted according to their connectivity, the most connected module will top the set. This is because that the most connected module affects many other modules at the allocation step, and thus it is better to allocate and fix it before others.

The goodness evaluation step is similar to GBSA with some modifications in the goodness measure. Since the area SD minimization is an objective like the TSV minimization, it is more realistic and effective to encounter it in the module goodness. So, the goodness measure is composed out of two components: the TSV based goodness and the area SD based goodness.

The area SD based goodness (GM_SD_1) is defined in Figure 3.10. In the figure, (I) is the module, Layer (I) denote the layer that contains I, and AVG is the average area of all layers. As explained in the figure, this goodness measure depends on the area of the layer that contains the module. If the layer area is close to the average area, the goodness will be high; the higher the layer area is the lower the goodness of the module will be. The significance of this measure is coming from its tendency to fix the modules in layers with small area at their locations, because moving them will increase the SD more and more. On the other hand, this measure assigns low goodness values to the modules in

layers with large area in order to increase the probability of moving them and consequently decrease the layer area which make it close to the average area.

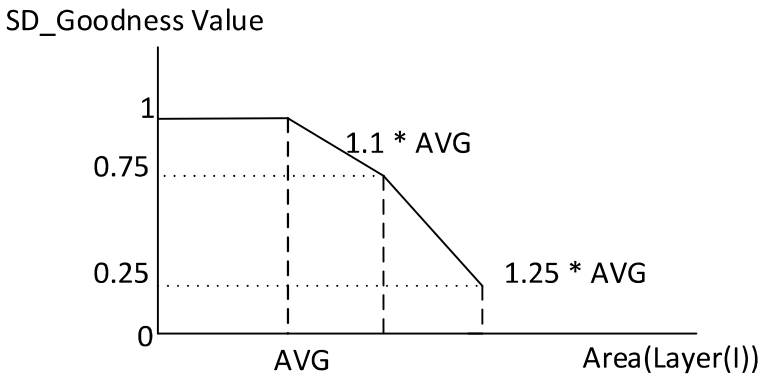

Figure 3.10: Graphical Representation of GM_SD_1 Measure.

For TSV based goodness we defined and tested three different measures: GM_TSV_1, GM_TSV_2, and GM_TSV_3. GM_TSV_1 is defined in subsection 3.2.1 and it is used in this approach without any modifications.

The second measure (GM_TSV_2) introduces the concept of net goodness. The goodness of a net is calculated from the actual TSVs ($TSVC_{net}$) that the net has, and the maximum number of TSVs required by a net (Max_TSV_{net}). For non-IO nets (nets with no IO pins) the maximum is the number of layers decremented by one (at least one module in the first layer and another one in the last layer), for IO nets the maximum is the number of layers (one module at least in the top layer and the IO pin in layer 0). eq. 17 formulates the goodness of a net. The goodness of a module in GM_TSV_2 is calculated as the average of the net goodness of all nets that contain the module.

$$Net\_Goodness = 1-(TSVC_{net}/ Max\_TSV_{net}) \quad \text{..... (eq. 17)}$$

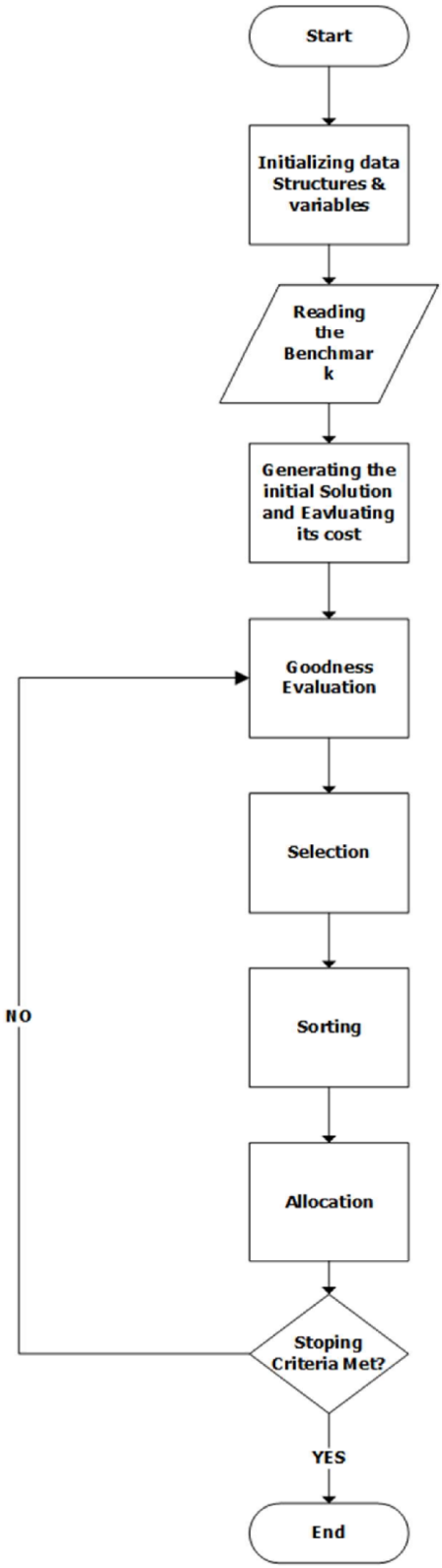

Figure 3.11: General Flow of SimE Algorithm.

The third measure (GM_TSV_3) is based on FM method. This measure associates five values with each module. Let S_i denotes the set of modules that are connected to module (i), D_i is the cardinality of S_i, and L denotes the number of layers. The first value is the maximum gain ($gmax_i$); this value represents the worst case of a module allocation where module (i) is assumed to be in the last layer and all other non-IO modules in S_i are located in the first layer. eq. 18 formulates this value. The second value is the minimum gain ($gmin_i$); this value represents the best case of a module allocation where the module (i) and all other non-IO modules in S_i are in the same layer. eq. 19 formulates this value. If the module is connected to an IO pin, then $gmax_i$ is incremented by the number of layers, and $gmin_i$ is incremented by one.

$$gmax_i = Di * (L\text{-}1) + B \quad \text{(eq. 18)}$$

$$gmin_i = -Di + B \quad \text{(eq. 19)}$$

Another two values are F_i and T_i. Let Layer (i) denotes the layer that contains the module i; F_i is calculated as the sum of the difference in layers between module (i) and all other modules in S_i. T_i is the number of modules in S_i that are located in Layer (i). F_i and T_i are formulated in eq. 20 and 21 respectively.

$$F_i = \sum_{j\ in\ S_i} |Layer(i) - Layer\ (j)| \quad \text{(eq. 20)}$$

$$T_i = \sum_{j\ in\ S_i,\ Layer(j)=Layer(i)} 1 \quad \text{(eq. 21)}$$

The final value is G_i; this value is simply the difference between F_i and T_i as shown in eq. 22.

$$G_i = F_i - T_i \quad \text{(eq. 22)}$$

The goodness value of module (i) in this method is calculated using G_i, $gmax_i$, and $gmin_i$ according to Figure 3.12. As shown in the figure, the lower values of G_i have goodness values close to one. Lower values of G_i mean that T_i is greater than F_i; in other words, the number of connected modules that are in the same layer of module (i) is larger

than the ones that are in other layers, consequently the lower values of G_i indicates that the module location is relatively good.

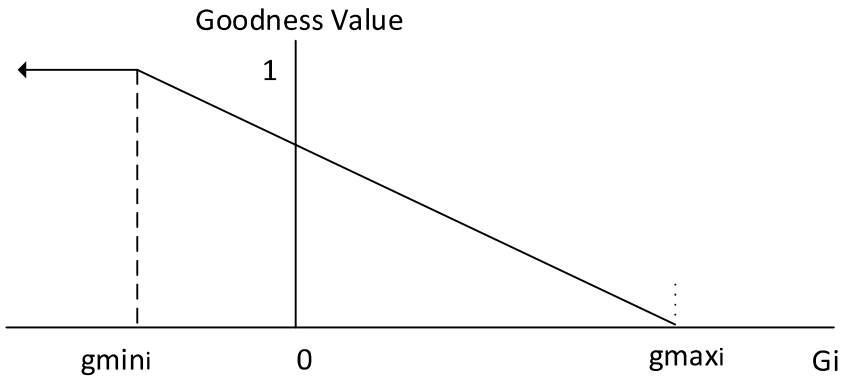

Figure 3.12: Graphical Representation of GM_TSV_3 Measure.

The overall goodness of a module is calculated by combining the SD based goodness measure (GM_SD_1), and one of the TSV based goodness measures; the combining is carried using eq. 8 to generate the total combined goodness (TCG) measure as in eq. 23.

$$TCG = Bg * GM\_TSV\_x + (1-Bg)* GM\_SD\_1 \ldots\ldots\ldots\ldots \text{(eq. 23)}$$

The last step in this approach is the allocation step. In this step all selected modules in PS are allocated to the corresponding layers. The allocation process starts from the top of PS. The core of this step is the criteria that determine to which layer a specific module should be assigned. Since the number of layers is limited, we can develop a greedy approach that allocates the module to the layer with minimum cost increment (maximum resultant TCO). Figure 3.13 summarizes this method.

```
Define ALLOCATION
    Input: Set PS, Number of layers L
    For all I in PS
        MIN=0, LA = 0
        For k = 1 : L
            Evaluate TCO after allocate I to layer K
            If (TCO > MIN)
                LA = k
                MIN = TCO
            End If
            Layer (I) = LA
            Cost Update
        End For
    End For
End Definition
```

Figure 3.13: Procedure for the Greedy Allocation Method.

The difference between the evaluation and the update steps is that the evaluation step only evaluates the resultant TCO without changing any of the cost variables or the current assignment state; the cost update takes place after completing the evaluation step and determining the best layer. Cost update step is the commitment or actual allocation of the module to the best layer specified in the evaluation step.

3.3 Simulation and Results

This Section explains the implementation and simulation of the adopted approaches along with the produced results. Also it discusses the results and compares it to some work in the literature. The implementation of all the work in this part was done using the C-programming language over the Windows OS.

Regarding the simulation there are a collection of parameters that are related to the simulation itself (Number of Runs, Number of iterations in each run), the number of layers, and the weight in multi-objective combining functions (Bc in eq. 16, Bg in eq. 23). Also there are some parameters that are related to the specific algorithm in use (e.g. T0 in SA). For the work in this part the main measures and results that reflect the quality

of the solution and highlight the trend of the approach are: the TSV Count, the COV, the TCO, and the average goodness of all modules. In the following two subsections, for each developed approach (GBSA, SimE) we summarized the simulation parameters and the significant results and measures. Also we discussed the produced results and we compared it against the SA algorithm and TS (Tabu Search) algorithm that have been applied by Sait, et al. [35]. Also we compared the developed approaches against each other.

3.3.1 GBSA: Simulation and results

In this part we simulated the GBSA algorithm for 15 runs, in each run the algorithm executed for 25000 or 80000 iterations. The considered number of layers is 4. The other used parameters in this approach are the same as the ones used by Sait, et al. [35]. These parameters are summarized in Table 3.2.

To mitigate the effect of the initial solution and the randomness nature of the algorithms, we considered and reported the average values of all measures. These average values are calculated form all the results that we got out of the 15 runs. This approach is a simulated annealing based approach, so it is better first to summarize the results of this approach and compare it against the SA results reported by Sait, et al. [35].

Table 3.3 tabulated the results of GBSA and SA For all benchmarks. From the table results we can observe two things. First, in terms of TSV Count the GBSA always outperform the standard SA, and this is because of introducing the Goodness concept that direct the algorithm to swap most of the time the bad modules while preserve the locations of the good modules. The second important thing is the execution time; since both algorithms are SA based; we compared the number of iterations that the algorithm take to converge, as we can see the GBSA takes 0.25-0.8 the number of iterations that SA takes to converge to the desired results; and this is a significant improvement in terms of the execution time. To highlight the trend of the GBSA algorithm, we plot the different measures of this algorithm for two circuits: n200 (small circuit) and frisc (large circuit).

Table 3.2: GBSA Simulation Parameters.

Parameter	Value
Bc (for combining the cost Functions)	0.8
T0 (Initial Temperature)	40
Alpha	0.98
Beta	1
M	50

Table 3.3: GBSA Simulation Results.

	TSV Count			TCO			No. of Iterations		
TB	SA	GBSA	GBSA/SA	SA	GBSA	SA/GBSA	SA	GBSA	GBSA/SA
n100	996	985	1.01	0.820	0.836	1.02	100000	25000	4
n200	2035	1921	1.06	0.795	0.808	1.02	100000	25000	4
n300	2133	2099	1.02	0.820	0.824	1.01	100000	25000	4
tseng	1091	1081	1.01	0.924	0.904	0.98	100000	80000	1.25
diffeq	1309	1221	1.07	0.930	0.904	0.97	100000	80000	1.25
elliptic	3683	2766	1.33	0.897	0.898	1.00	100000	80000	1.25
frisc	3502	3075	1.14	0.904	0.9	1.00	100000	80000	1.25
pdc	6398	6287	1.02	0.865	0.862	1.00	100000	80000	1.25

The average goodness of n200 and frisc circuits is shown in Figures 3.14 and 3.15 respectively. As we can see in the figures, at the initial stage the modules have low goodness values, but as the algorithm progresses the modules are assigned to more suitable layers and the overall goodness improved.

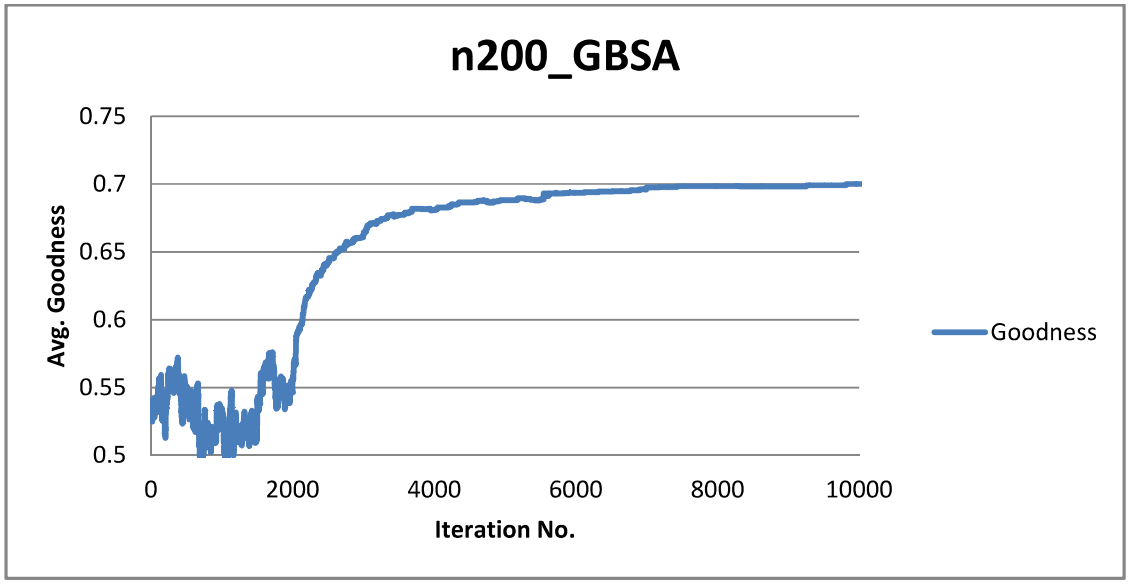

Figure 3.14: Change in Average Goodness of n200 circuit in GBSA with Iteration.

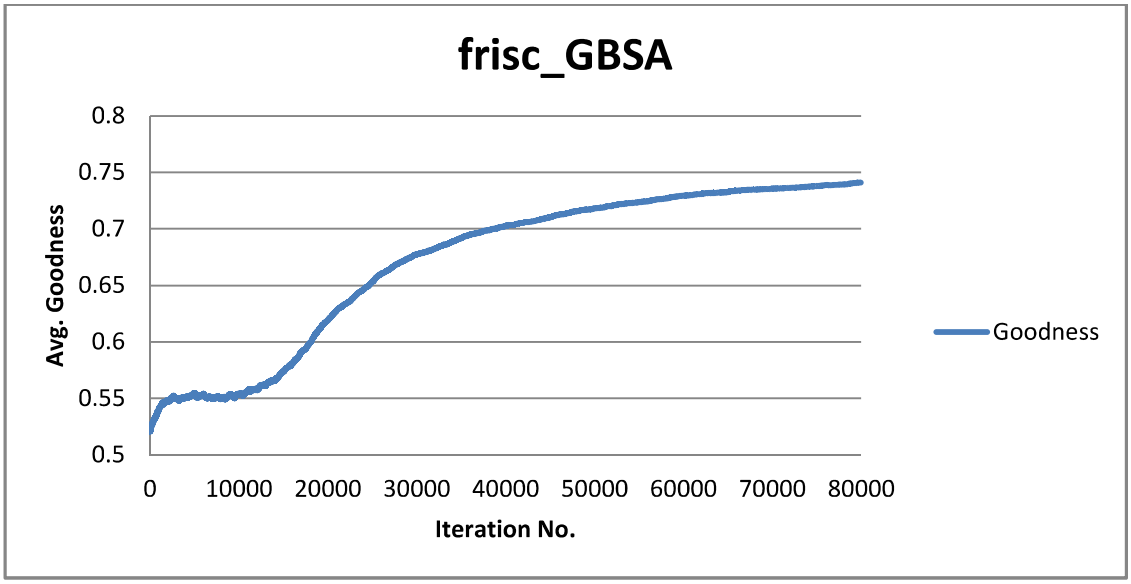

Figure 3.15: Change in Average Goodness of frisc circuit in GBSA with Iteration.

Figures 3.16 and 3.17 show the TSV Count vs. iteration plots for the same circuits. As we can see from the figures, at the early iterations the TSV Count values

fluctuates because of the acceptance of bad solutions (because T is high); but as the algorithm progresses it become greedy and only better solutions are accepted.

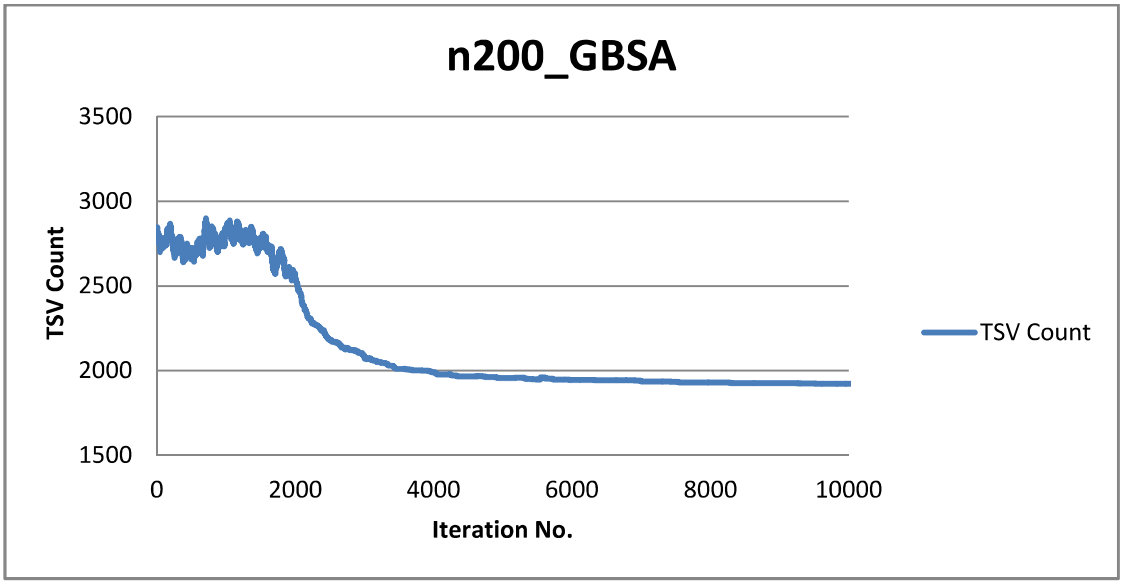

Figure 3.16: Change in TSV Count of n200 circuit in GBSA with Iteration.

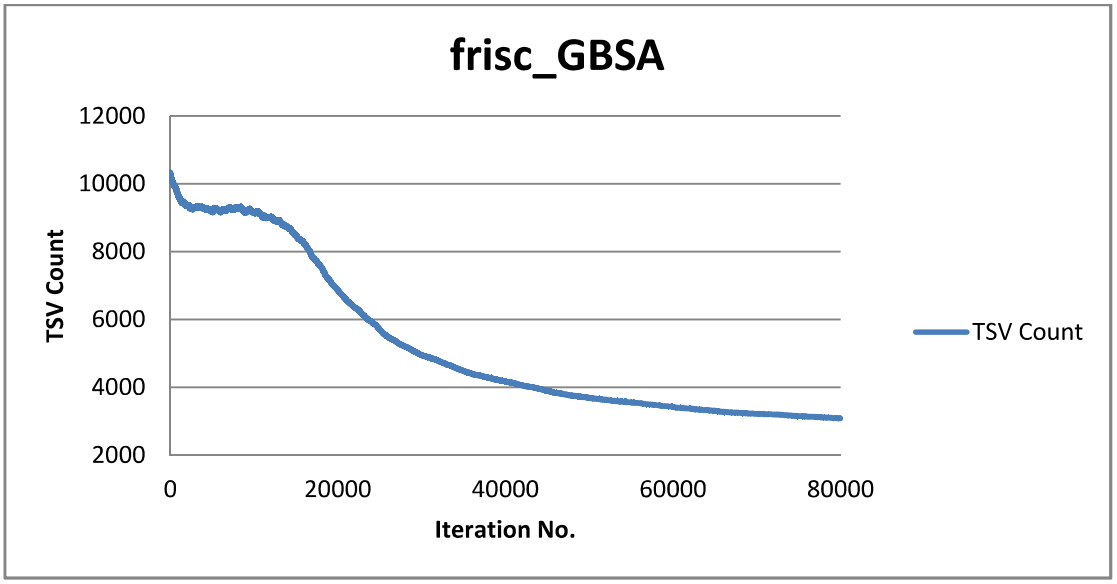

Figure 3.17: Change in TSV Count of frisc circuit in GBSA with Iteration.

The combined objective (TCO) of both circuits is shown in Figures 3.18 and 3.19. As we can notice in the figures, the TCO initially is low and this is because the higher values of the TSV Count and the area SD; as the algorithm proceeds, the TSV Count and SD decrease and consequently the TCO is improved.

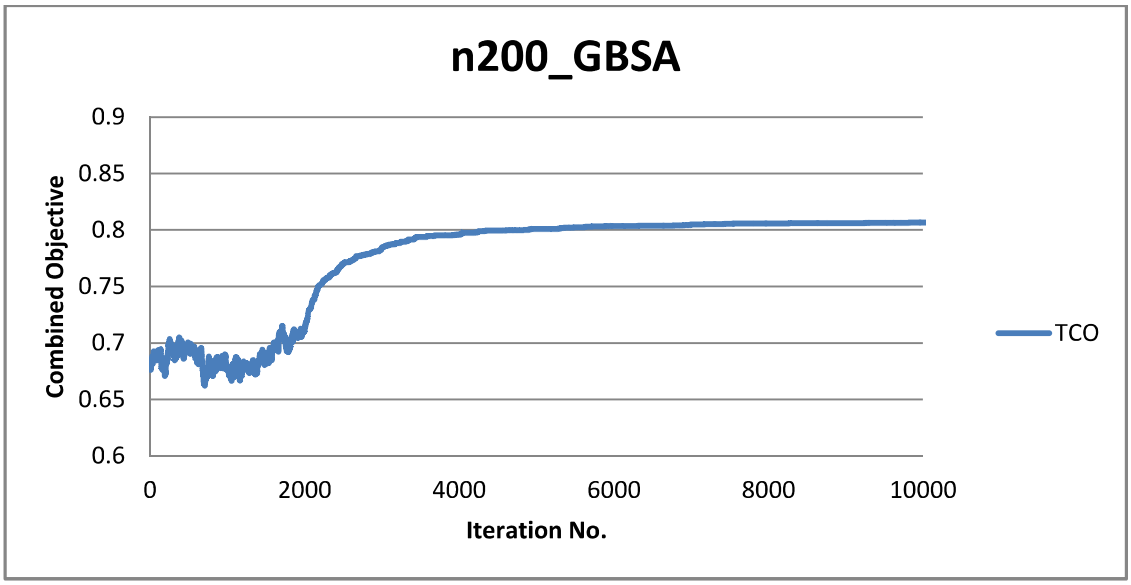

Figure 3.18: Change in TCO of n200 circuit in GBSA with Iteration.

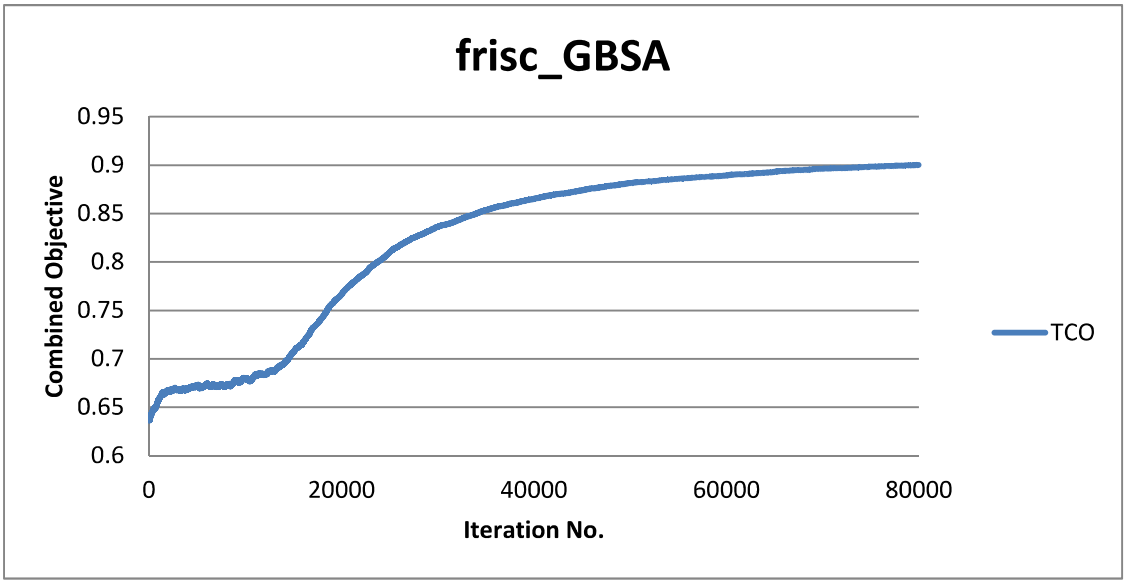

Figure 3.19: Change in TCO of frisc circuit in GBSA with Iteration.

From the above results, we can conclude that incorporating the goodness concept in the SA improves its outcome especially the overall TSV Count. Also this modification speeds up the algorithm which makes it converge to the desired results in short execution time (less number of iterations).

3.3.2 SimE: Simulation and results

In this part we simulated the SimE algorithm using the different proposed goodness measures. For each goodness measure the algorithm has been simulated for 15 runs, in each run the algorithm executed for 1000 iterations. The considered number of layers is 4. Bc is tuned to 0.8 and Bg is tuned to 0.

To mitigate the effect of the initial solution and the randomness nature of the algorithms, we considered and reported the average values of all measures. These average values are calculated using all the results that we got out of the 15 runs.

Since we developed and proposed different goodness measures that can be incorporated in the SimE (GM_TSV_1, GM_TSV_2, GM_TSV_3), it is more significant to summarize the results of the algorithm for each measure and compare the produced results against each other, then we compare the SimE results with the GBSA reported results in the previous subsection and the TS results that are reported by Sait, et al. [35].

Table 3.4 tabulated the results of SimE for the different goodness measures. From the table we can observe that the different measures give close results for some (small) circuits and variant results for other (large) circuits. Also we can see that the GM_TSV_1 measure produces the best results among the three measures. This measure is the simplest one and still it is the most powerful and efficient one. The reason that makes this measure as effective as this is its ability to reflect precisely the goodness of the location of the module in the current assignment. To highlight the trend of the SimE algorithm, we plot the different measures of this algorithm for two circuits: tseng (small circuit) and elliptic (large circuit).

The average goodness of tseng and elliptic circuits is shown in Figures 3.20 and 3.21 respectively. As we can see in the figures, at the initial stage the modules have low goodness values, but as the algorithm progresses the modules are assigned to more suitable layers and the overall goodness improved. The GM_TSV_1 measure converges to the highest avg. goodness value. The other goodness measures have a small fluctuating avg. goodness values even at the later iterations; the reason behind this is the incapability

of these measures to reflect the goodness of the modules in a precise and comprehensive way. These figures implied that the GM_TSV_1 measure is the best measure among the three as we stated earlier.

Table 3.4: Simulation Results of SimE Approach.

	TSV Count			TCO		
TB	GM_TSV_1	GM_TSV_2	GM_TSV_3	GM_TSV_1	GM_TSV_2	GM_TSV_3
n100	957	955	1005	0.851	0.821	0.842
n200	1922	1973	2012	0.823	0.815	0.808
n300	1965	2236	2173	0.847	0.823	0.828
tseng	983	979	975	0.934	0.915	0.935
diffeq	1209	1214	1492	0.932	0.914	0.904
elliptic	2376	2642	3181	0.940	0.934	0.920
frisc	2454	2479	2835	0.938	0.918	0.909
pdc	2426	3861	4605	0.943	0.93	0.890

Figures 3.22 and 3.23 show the TSV Count vs. iteration plots for the same circuits. As we can see from the figures, at the early iterations the TSV Count values are large but after only small number of iterations the TSV count value is reduced significantly and converges quickly to the desired final result. Also we can observe that the GM_TSV_1 always give the minimum TSV Count.

The Combined objective (TCO) of both circuits is shown in Figures 3.24 and 3.25. As we can notice in the figures, the TCO initially is low and this is because the higher values of the TSV Count and the area SD; as the algorithm proceeds, the TSV Count and SD decrease quickly and consequently the TCO is improved after small number of iterations.

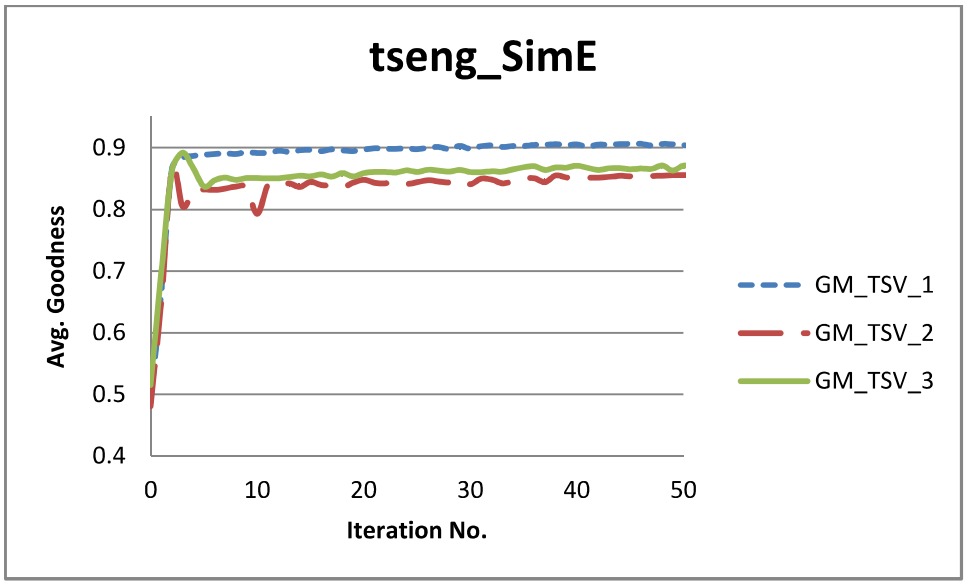

Figure 3.20: Change in Average Goodness of tseng circuit in SimE with Iteration.

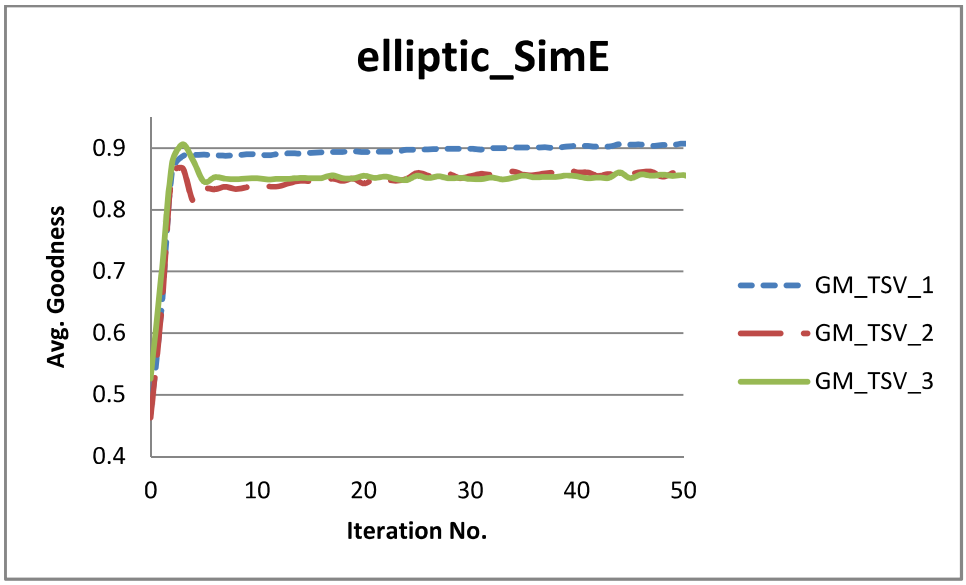

Figure 3.21: Change in Average Goodness of elliptic circuit in SimE with Iteration.

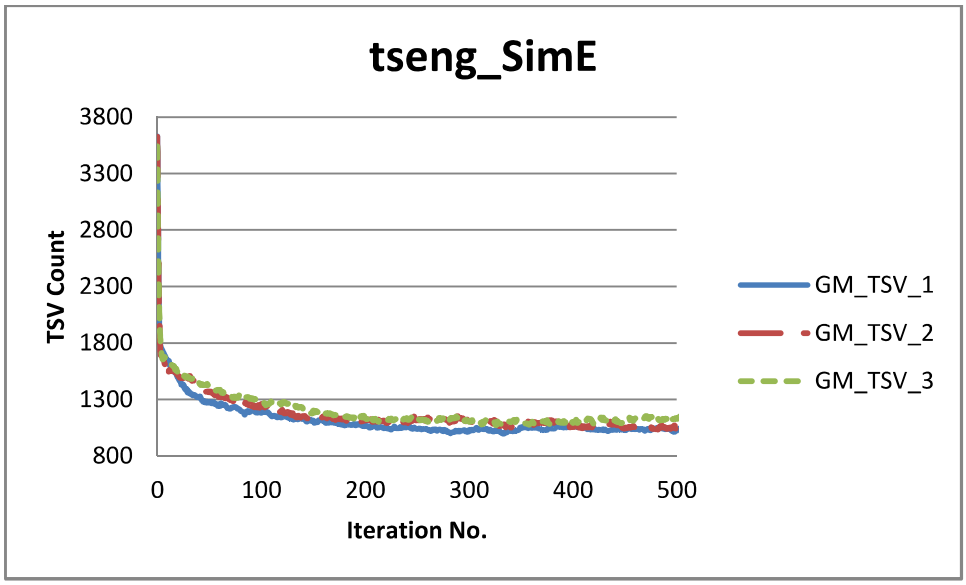

Figure 3.22: Change in TSV Count of tseng circuit in SimE with Iteration.

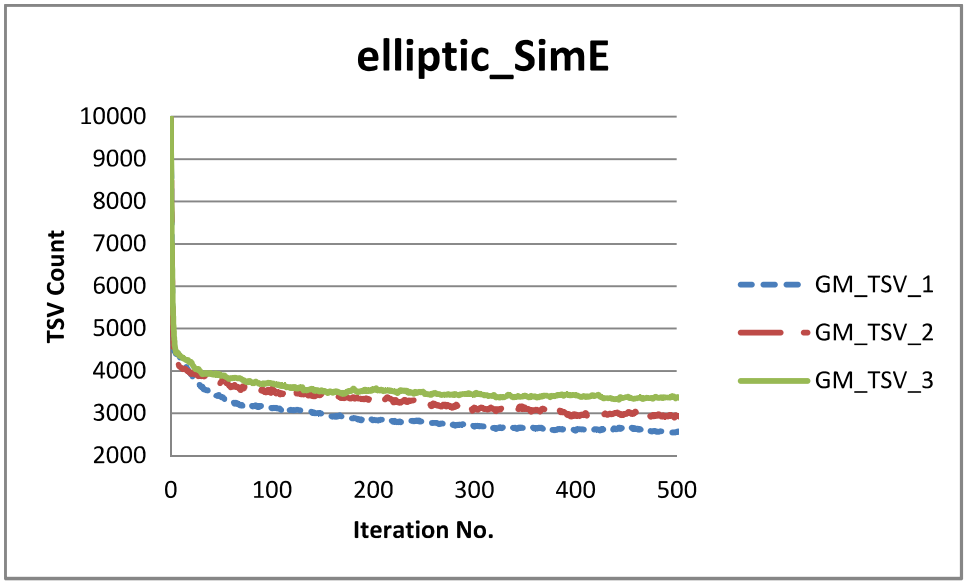

Figure 3.23: Change in TSV Count of elliptic circuit in SimE with Iteration.

From the previous results we can conclude that SimE performs well in optimizing the problem in question, and also the suitable and efficient goodness measure that can be used effectively in the SimE is the GM_TSV_1 measure. Now, to highlight the strength of SimE algorithm in solving this problem, we will conclude this subsection by comparing the SimE algorithm that utilize the GM_TSV_1 measure with the GBSA (this

algorithm outperform the SA- see the previous subsection) and TS algorithm applied by Sait, et al. [35].

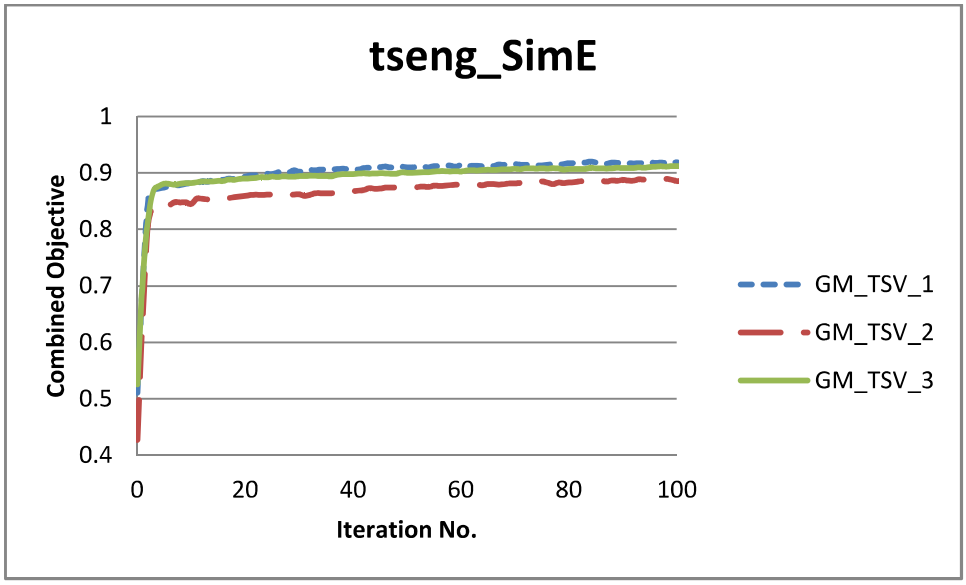

Figure 3.24: Change in TCO of tseng circuit in SimE with Iteration.

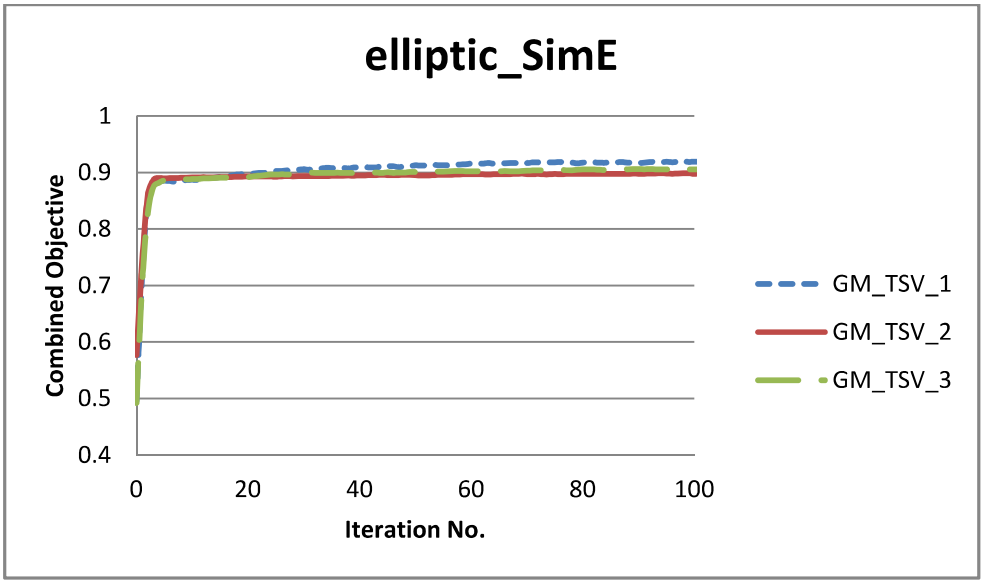

Figure 3.25: Change in TCO of elliptic circuit in SimE with Iteration.

Table 3.6 summarizes all the results from SimE, GBSA, and TS algorithms. Table 3.7 summarizes the percentage improvement of SimE over other methods

Table 3.5: Comparison between SimE and other approaches.

	TSV Count			TCO		
TB	SimE	GBSA	TS	SimE	GBSA	TS
n100	957	985	991	0.851	0.836	0.821
n200	1922	1921	2076	0.823	0.808	0.791
n300	1965	2099	2210	0.847	0.824	0.813
tseng	983	1081	1568	0.934	0.904	0.910
diffeq	1209	1221	1986	0.932	0.904	0.893
elliptic	2376	2766	4365	0.940	0.898	0.870
frisc	2454	3075	4442	0.938	0.9	0.870
pdc	2426	6287	7296	0.943	0.862	0.846

Table 3.6: SimE improvements over different approaches.

	TSV Count Improvement		TCO Improvement	
TB	Over GBSA	Over TS	Over GBSA	Over TS
n100	103%	104%	102%	104%
n200	100%	108%	102%	104%
n300	107%	112%	103%	104%
tseng	110%	159%	103%	103%
diffeq	101%	164%	103%	104%
elliptic	116%	184%	105%	108%
frisc	125%	181%	104%	108%
pdc	259%	301%	109%	111%

As shown in the above tables, the SimE algorithm outperforms both of the other algorithms. The main reasons behind this are two things. The first thing is the introducing of a suitable goodness measure that reflects the real goodness of the current location of the module. GBSA also utilizes the same goodness measure; but the main difference between SimE and GBSA is the second source of strength in SimE; this reason is that

GBSA allocation is a random swapping between two selected modules; on the other hand, the SimE allocation step is performed in a way such that the module is assigned to the layer that maximizes the overall combined objective. These two reasons are the main sources of strength of the SimE algorithm when applied to solve any optimization problem.

CHAPTER 4

TSV AND WIRELENGTH OPTIMIZATION IN CELL DESIGN

The placement step in the physical design stage has an important and critical impact on the overall design performance. In the 2D cell design, the most important thing has to be handled is the total Wirelength. In 3D design, if the placement of the modules is taking place without any pre-steps of layers assignment, then the placement problem should handle both the total wirelength and the required number of TSVs. This optimization problem is a multi-objective one that targets minimizing the TSV Count and total WL simultaneously and takes into account some constraints that are related to the 3D design requirements; one of these important constraints is the balancing of the utilization (percentage of occupied locations) of all layers in the 3D chip. The scope of this work is the cell design style where the modules in the circuit have comparable areas.

In this part of our work we specified and applied the SimE algorithm to minimize the required number of TSVs and the lateral WL. The developed solution has been tested against a group of benchmarks that have different sizes (in terms of modules and nets) and lower modules area standard deviation (modules areas are close to each other which make it suitable for the scope of this work). The rest of this Chapter is organized as follow. The first Section explains the problem in question along with the state representation adopted and the formulated cost function. After that, the developed solution and the detailed steps of the applied algorithm are explored in the second Section. Finally, in the third Section the results are summarized and discussed.

4.1 Problem Formulation

This part of our work investigates the placement problem in 3D cell design. The objective of this work is to minimize the required TSV Count (number of TSVs), and

minimize the overall WL. Also in this problem we constrain the resultant solution to utilize all design layers.

The placement problem is defined as follows:

Given the following inputs:

1) A set of design layers (numbered from 1 to L)
2) A set (V) of M modules
3) The set of IO pins (IOC pins)
4) The netlist (a list of all nets in the circuit (N nets))

We have to determine the (x, y, z) coordinates for each module and each required TSV such that the following objectives are minimized:

1) TSV Count
2) Overall WL

And the following constrain is satisfied: the layers utilization COV should be less than 15%.

In solving this problem we assume the following:

1) A F2B stacking style is used
2) A via first TSV is considered
3) The IO pad is located at the bottom below all layers (at layer 0), this layer has zero area
4) Each net utilizes one exclusive TSV between two consecutive layers
5) To connect the modules of a single net in two consecutive layers, the TSV will be assigned to the top layer; e.g. if net x has modules in layer 1 and layer 2, then a TSV will be assigned to layer 2
6) All TSVs have the same area
7) The layers have the same area and same aspect ratio; also the outline of the layers is fixed and predetermined

4.1.1 State Representation

Most of the structures that are used to represent the different components are the same as in Chapter 3. The 3D chip consists of a number of layers that is stacked above the IO pad. Since in this part we are concerned with assigning the modules to the corresponding layers and to the corresponding locations within the layer; the 3D chip is represented as an **ordered** list of layers numbered from 0-L where layer 0 represent the IO pad and can contain only IO pins. Each layer (1-L) is represented as 2D array of cells, where each cell can handle one module/TSV. The 2D array dimensions are xmax (length) and ymax (width), thus the layer is composed out of xmax rows and ymax columns, where the intersection of any row with any column is a cell located at (x, y) where x: row number, and y:column number. Layer structure is explained in Figure 4.1.

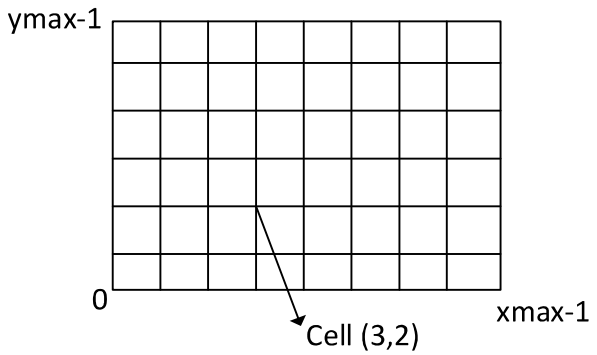

Figure 4.1: Layer Representation in 3D Placement Problem.

The layer outline (determined by xmax and ymax) and the aspect ratio are fixed and predefined; the values of xmax and ymax depend on the size of the circuit (number of modules).

The netlist is presented as an array of linked lists as in Chapter 3; also the connections between modules are captured as a 2D square matrix as explained in the previous Chapter.

4.1.2 Cost Formulation

In this multi-objective problem, there are two components contribute to the total cost: the TSV Count and the Overall WL. The TSV Count component is defined exactly as in subsection 3.1.2.

The overall WL is computed as the sum of the wirelength of all nets. The wirelength of a net is calculated as the sum of the net WL in all layers. The WL of a net at specific layer is calculated as the SP of the box that encloses all the modules and the TSV of that net in this layer along with the TSV in the upper layer. In other words, the TSV in the upper layer is projected into the lower layer, and then a box that encloses all the modules, the TSV in the layer and the projected TSV is drawn. The WL is the SP of this box. Now, to formulate the previous values, let WL denotes the overall wirelength of the current placement, WL_n denotes the total WL of net (n) in all layers, and $WL_{n,l}$ denotes the wirelength of net (n) in layer (l). These values are formulated in eq. 24-26.

$$WL = \sum_{n=1}^{N} WL_n \quad \text{(eq. 24)}$$

$$WL_n = \sum_{l=1}^{L} WL_{n,l} \quad \text{(eq. 25)}$$

$$WL_{n,l} = \text{Length(BOX)+Width(BOX)} \quad \text{(eq. 26)}$$

To clarify eq. 26, consider the following example shown in Figure 4.2. In this example a net (n) has 2 modules and one TSV in the first layer, and another TSV and module in the second layer. To calculate the WL of this net in the first layer, we consider only the modules and TSV at the first layer (M1, M2, and TSV) and the TSV from the second layer (TSV2). First, TSV2 is projected from its (x, y) location in the second layer to the same (x, y) in the first layer. Then the enclosed box is drawn as seen in the figure (Red rectangle). $WL_{n,l}$ is calculated as the SP of the red rectangle.

The minimum WL value (WL_MIN) occurs when each net has all of its modules close to each other in the same layer. In this scenario, the enclosed box of each net is a rectangle with SP equal to $\left\lceil \frac{No\ of\ modules}{2} \right\rceil$; Figure 4.3 explains this case. As shown in the

figure, the SP of the rectangle that enclose all modules (encloses all modules coordinates) is 4; this is equal to $\left\lceil \frac{7}{2} \right\rceil = 4$. The maximum WL (WL_MAX) is assumed as the WL of the initial placement.

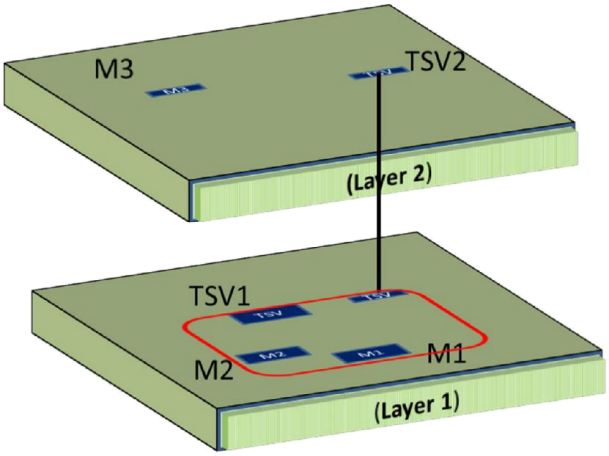

Figure 4.2: Example of WL.

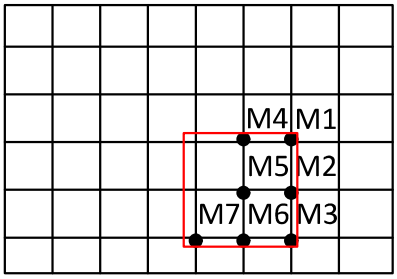

Figure 4.3: WL_MIN of a net (Example).

The membership function of the TSV Count is the same as in the previous Chapter; WL membership function is explained in Figure 4.4.

The above membership functions are combined according to eq. 8 to produce the total combined objective (TCO) as shown in eq. 27. Considering the total combined objective, the problem under investigation can be a viewed as an optimization problem with the target of maximization the total combined objective (TCO).

TCO = Bc * Normalized (TSV Count) + (1-Bc)* Normalized (WL)..... (eq. 27)

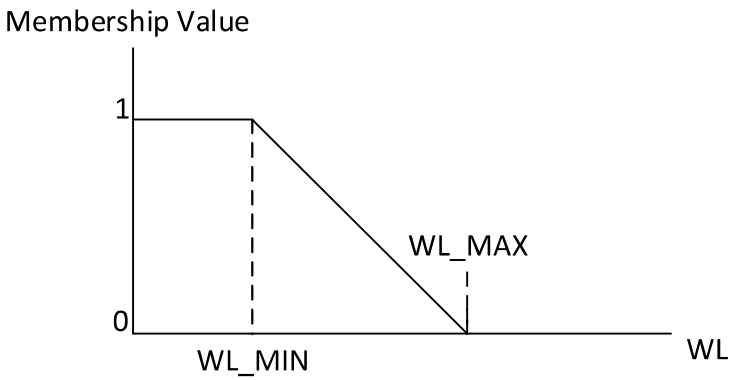

Figure 4.4: Membership Function of WL.

4.1.3 Benchmark Circuits

In order to test the quality of our solution, we have to test it on a collection of different well-known Benchmark Circuits. We selected a set of five Benchmark Circuits that are differ in the number of nodes, the number of IO pins, the number of nets, and have a low modules area SD (SD = 1 only). These Benchmarks are Available at: http://vlsicad.cs.binghamton.edu/benchmarks.html Table 4.1 summarizes the main characteristics of the used Benchmarks.

Table 4.1: Characteristics of Benchmark Circuits.

Name	No. of Modules	No. of IO pins	No. of Nets	Modules Area SD
tseng	2417	174	2295	1
diffeq	3024	103	2985	1
elliptic	6831	245	6717	1
frisc	7024	136	6908	1
pdc	7609	56	7569	1

4.2 Developed Approach

The problem under investigation is a 3D placement problem where the modules are not assigned to the corresponding layers in any pre-step; this means that the total number of required TSVs is not known yet. Thus this is a multi-objective hard problem that targets the minimization of both the TSV Count and the Total WL. The modules and TSVs have to be placed at their near optimum locations; and at the end, each module and each TSV should be associated with an exclusive (x, y, z) tuple. To solve this problem we adapted and customized the SimE algorithm that incorporates the FD algorithm in the allocation step.

In the placement problem we are required to place both the modules and the TSVs. Since the number of TSVs in each iteration is not known at the beginning; the allocation stage is performed in two ordered steps, the first step is the allocation of the modules; this step determines the required number of TSVs. After this step, each TSV is bounded to a specific net and a specific layer (z). The second step is to allocate each TSV within its z layer.

The general flow of the adapted SimE algorithm is shown in Figure 3.11. After initializing the required variables and data structures and reading the Benchmark into the desired data structures, the initial solution is generated by first assigning the modules to their (x,y,z) location on a random basis; then the required TSVs is assigned to their (x, y) location. The cost (TSV Count, WL) of this solution is calculated along with the values of TSV_MAX, TSV_MIN, and WL_MIN. The remaining steps will be executed in a repeated manner until the stopping criteria is met; the stopping criteria is either reaching the predefined maximum number of iterations or when the overall combined cost remains almost constant for a specific number of iterations.

Since the number of TSVs (in the current iteration) depends on the placement of the modules; at the beginning of the algorithm a list of TSVs is created. The number of TSVs in the list is equal to the TSV_MAX. Each TSV is associated to one net-layer combination. Since not all TSVs in the list are required (valid) each TSV is associated

with a status that indicates whether the TSV is **active** or not; only active TSVs are considered in the allocation step.

Since both of the modules and the active TSVs are considered in the allocation step, the goodness evaluation step and the selection step also consider both of them. Both of the evaluation and selection steps take place before the allocation, so these steps consider the active TSVs from the previous iteration. The selection step is carried out exactly as in Chapter 3; the only modification here is that we also perform this step on the TSVs. So at the end of the selection step, two sets are generated: PS (Selected modules) and PSTSV (Selected TSVs).

The evaluation step evaluates the goodness of the modules and the active TSVs. Since in the placement problem there are two objectives; the module goodness measure (MODULE_GOODNESS) consists of two components: the TSV based goodness and the WL based goodness. The TSV based goodness (GM_TSV_1) is same as the one defined in Figure 3.8.

WL based goodness (GM_WL_1) is defined in Figure 4.5. As explained in the figure, the WL based goodness of a module is the average of the goodness of the nets that contain the module. The goodness of each net is the optimum WL (minimum WL) of a net over the actual WL. The lower the actual WL of a net is the higher its goodness will be and consequently the higher the module goodness will be.

The overall goodness of a module is calculated by combining the WL based goodness measure (GM_WL_1), and the TSV based goodness measure (GM_TSV_1); the combining is carried out using eq. 8 to generate the total combined goodness (TCG) measure in eq. 28.

TGO = Bg * Normalized (GM_TSV_1) + (1-Bg)* Normalized (GM_WL_1)…. (eq. 28)

Define GM_WL_1

Input: module *I*

Output: WL_Goodness [*I*]

Let *S* denote the set of all nets that has I as an element

Let W_n denotes the actual WL of net n, and O_n the minimum WL of net n

Initialize $Tot = 0$, N = Cardinality(S)

For all n in S

$Tot = Tot + (O_n / W_n)$

End For

WL_Goodness [*I*] = Tot/N

End Definition

Figure 4.5: Description of GM_WL_1 Measure.

For the goodness measure of TSVs, only the WL based goodness is considered, because each TSV is independent and has nothing to do with the overall TSV Count. Since the TSV is associated with only one net, only this net is considered in the goodness of the TSV. Also since the TSV location affects the WL of the net in the layer where it is located, and in the layer below it, the WL of the net in both layers are considered. The goodness measure of a TSV (TSV_GOODNESS) is defined as the ratio of the WL_{before} to WL_{after}. WL_{before} is the WL of the net without considering the TSV, while WL_{after} is the WL of the net considering the TSV. Let *i* denotes the TSV, *n* denotes the corresponding net, and l denotes the layer (i). Using the parameters defined in eq. 25 and 26, we can formulate TSV_GOODNESS as follows:

$$WL_{n\ before} = WL_{n,l\ before} + WL_{n,l-1\ before} \quad \text{(eq. 29)}$$

$$WL_{n\ after} = WL_{n,l\ after} + WL_{n,l-1\ after} \quad \text{(eq. 30)}$$

$$TSV\_GOODNESS = WL_{n\ before} / WL_{n\ after} \quad \text{(eq. 31)}$$

Higher value of WL after considering the TSV means lower TSV goodness value. The only left step in the flow is the allocation step. As we mentioned earlier this step is carried out into two phases: the modules allocation phase and then the TSV allocation

phase. Two allocation schemes have been developed in this work: the FD approach and the Greedy-Random (GR) approach.

4.2.1 FD allocation approach

In the FD allocation approach, the modules allocation phase allocates all the modules in the PS set one after another. Figure 4.6 explained the allocation of a single module (FD_MODULE_ALLOCATION).

```
Define FD_Module_Allocation
        Input: Module I
        Calculate the zero-force location (x, y, z) of I using FD algorithm
        If (z violates the utilization constraint)
                z = the nearest layer that satisfy the constraint
        If ((x, y, z) is empty)
                Allocate the module at (x, y, z)
                Mark (x, y, z) as occupied
        Else
                Find the nearest free location (x1, y1, z1)
                Allocate the module at (x1, y1, z1)
                Mark (x1, y1, z1) as occupied
        End If
End Definition
```

Figure 4.6: FD Module Allocation Algorithm.

As explained in the figure, the module is allocated to its zero-force location which is found using the FD algorithm (see Section 1.4). Only the already allocated modules (not-selected) and the active not-selected TSVs are considered in the FD algorithm. If the calculated layer violates the layer utilization constraint (this happens when the layer has occupied locations much larger than the layers average of occupied locations), then the algorithm finds the nearest layer that satisfy the utilization constraint. After that the module is allocated to the corresponding location; if the location is occupied then the module will be assigned to the nearest free location. Once the module is located, its (x,y,z) is updated and the location is flagged as occupied.

TSV allocation phase takes place after allocating all the modules, at the beginning of this phase only the required TSVs are active and consequently this phase allocates all the active TSVs in the PSTSV one after another. This phase is similar to the module allocation phase with two exceptions. The first one is that since this phase takes place after modules allocation, the FD algorithm calculates the location of the TSV considering all the modules and the active not-selected TSVs. The second important thing is that since the layer of the TSV is fixed and predefined, the FD algorithm calculates only the (x,y) location of the TSV within that layer. If the layer is full, the TSV will be placed in the location of the nearest module to its calculated location and the module is placed all over again using the module allocation method.

4.2.2 GR allocation approach

The GR allocation approach is a greedy-random approach that allocates the modules and TSVs in a greedy-random manner. By greedy we mean that the algorithm allocates the module/TSV to the location that produces the minimum cost increment (maximizes the overall TCO). Since the number of locations is not small like the case in Chapter 3, the greedy allocation can't cover all possibilities because they are large and exploring them is not feasible in term of execution time. This approach is greedy to some extent and it avoids the brute force infeasible case by introducing some form of randomness.

As in the FD approach, the modules allocation phase allocates all the modules in the PS set one after another. Figure 4.7 explained the allocation of a single module in the GR approach (GR_MODULE_ALLOCATION).

As explained in the figure, the first step in this approach is to find the best layer that minimizes the TSV Count and at the same time satisfy the utilization constraint (greedy approach). Then the rectangle that encloses all the modules that belong to all nets that contain the module in question is drawn. This rectangle narrows the range of locations that the algorithm has to search to find the best location of the module. The rectangle is defined by the lower left corner (x0, y0) and the top right corner (xt, yt). After that the algorithm search for random free locations within the enclosed rectangle,

the location that maximizes the TCO is selected (greedy). If the rectangle encloses no free locations then the algorithm selects the nearest free location to the borders of the rectangle. If the layer is full, the nearest available layer is selected and the process start all over again. Once the module is located, its (x,y,z) is updated and the location become occupied.

TSV allocation phase takes place after allocating all the modules; at the beginning of this phase only the required TSVs are active and consequently this phase allocates all the active TSVs in the PSTSV one after another. This phase is similar to the module allocation phase with one exception. Since the layer of the TSV is fixed and predefined, the GR algorithm find only the (x,y) location of the TSV within that layer. If the layer is full, the TSV will be placed in the location of another module and the module is placed all over again using the module allocation method.

```
Define GR_Module_Allocation
    Input: Module I
    Calculate the layer (z) that minimizes the TSV Count and satisfy the utilization constraint
    Calculate the corners (x0, y0) and (xt, yt) of the Enclosed Rectangle
    Ry = (yt-y0)/10, Rx = (xt-x0)/10
    Best_Cost =0
    Starting from (x0, y0)
    While ((x2, y2) is not reached)
        Generate Random r1 [1, Rx] and Random r2 [1, Ry]
        xr = x0 + r1, yr = y0 + r2
        If ((xr, yr, z) is within the rectangle and is empty)
            Evaluate TCO after allocate I to (xr, yr, z)
            If (TCO > Best_Cost)
                Best_Cost = TCO
                x = xr
                y = yr
            End If
        End If
    End While
    If (Free location not found)
        Find the nearest free location to the rectangle borders
    End If
    Allocate the module at (x, y, z)
    Mark (x, y, z) as occupied
End Definition
```

Figure 4.7: GR Module Allocation Algorithm.

4.3 Simulation and Results

This Section explains the implementation and simulation of the adopted approaches along with the produced results. Also it discusses the results and compares them to each other. The implementation of all the work in this part was done using the C-programming language over the Windows OS.

Regarding the simulation there are a collection of parameters that are related to the simulation itself (Number of Runs, Number of iterations in each run), the number of layers, the dimensions of each layer (xmax, ymax), and the weight in multi-objective combining functions (Bc in eq. 27, Bg in eq. 28). For the work in this part the main measures and results that reflect the quality of the solution and highlight the trend of the approach are: the TSV Count, the overall WL, the TCO, and the average goodness of all modules. For each developed approach (FD, GR) we summarized the simulation parameters and the significant results and measures. Also we discussed the produced results and we compared it against each other.

In this part we simulated the SimE algorithm using the different proposed allocation methods (FD, GR). For each method the algorithm has been simulated for 10 runs, in each run the algorithm executed for 250 iterations. The considered number of layers is 4. The dimensions of the layers are computed at the beginning of the simulation according to eq. 32 and 33. As we can see from the equations, the dimensions of the layer is different for the different benchmarks, and this makes sense because of the difference in the number of modules and the number of required TSVs in each circuit. The other used parameters in this approach are summarized in Table 4.2.

$$\text{xmax} = \sqrt{\frac{(number\ of\ modules + 1.2 * Initial\ TSV\ Count)}{Layers}} \quad \text{..... (eq. 32)}$$

$$\text{ymax} = \sqrt{\frac{(number\ of\ modules + 1.2 * Initial\ TSV\ Count)}{Layers}} \quad \text{..... (eq. 33)}$$

To mitigate the effect of the initial solution and the randomness nature of the algorithms, we considered and reported the average values of all measures. These average

values are calculated form all the results that we got out of the 10 runs. Since we developed and proposed different allocation schemes that can be incorporated in the SimE (FD, GR), it is more significant to summarize the results of the algorithm for each scheme and compare the produced results against each other.

Tables 4.2 and 4.3 tabulated the results of SimE for the different allocation schemes. From the table results we can observe that the FD approach outperforms the GR approach for most of the circuits in term of the TSV Count and the overall WL. Also we can notice from the numbers the FD approach speed compared to the GR method.

The main reason behind this is the nature of the FD approach itself. FD algorithm determines the best location (in terms of WL and TSV Count) of the element considering all other elements that are connected to it. This also is a direct numerical calculation that requires small execution time. On the other hand, GR approach is a greedy random based approach that only determines a wide range of locations that are suitable for the element, and then it evaluates a random number of locations and selects the one that gives the minimum cost. Since the algorithm does not examine all the possible locations within the range, there is a high probability (especially for the large circuits) that the best locations are not examined. Also since this algorithm conducts an evaluation for a number of locations each time an element is located, the overall execution time of this algorithm is relatively high.

Table 4.2: Simulation Results of Placement Problem.

			TSV Count		WL		TCO		Time (sec)	
TB	xmax	ymax	FD	GR	FD	GR	FD	GR	FD	GR
tseng	46	46	675	1345	18823	30660	0.890	0.779	771	2355
diffeq	53	53	832	1263	25384	49158	0.895	0.778	1296	3667
elliptic	79	79	2288	3139	129968	154823	0.832	0.784	8879	18205
frisc	80	80	3525	2236	175250	158502	0.772	0.808	10656	20820
pdc	84	84	1811	5510	134267	302891	0.882	0.691	15438	37172

Table 4.3: FD improvement over GR.

TB	TSV Count	WL	TCO	Time (sec)
tseng	1.99	1.63	1.14	3.05
diffeq	1.52	1.94	1.15	2.83
elliptic	1.37	1.19	1.06	2.05
frisc	0.63	0.90	0.96	1.95
pdc	3.04	2.26	1.28	2.41

To highlight the trend of the SimE algorithm and its different allocation methods, we plotted the different measures of this algorithm for two circuits: tseng (small circuit) and pdc (large circuit).

The average goodness of tseng and pdc circuits is shown in Figures 4.8 and 4.9 respectively. As we can see in the figures, at the initial stages the modules have low goodness values, but as the algorithm progresses the modules are assigned to more suitable locations and the overall goodness improved. The GR approach avg. goodness converges quickly because the algorithm has a little improvement on the overall objective in the later stage. In the FD method, the avg. goodness is improved gradually and at the later iterations it reaches higher values than the GR; this is because of the FD capability to improve the overall objective as the execution proceeds.

Figures 4.10 and 4.11 show the TSV Count vs. iteration plots for the same circuits. As we can see from the figures, at the early iterations the TSV Count values are large but after only small number of iterations the TSV count value reduced significantly and converges quickly to the desired final result. Also we can observe that the FD approach always gives the minimum TSV Count, while the GR improves the TSV Count slowly in the later iterations.

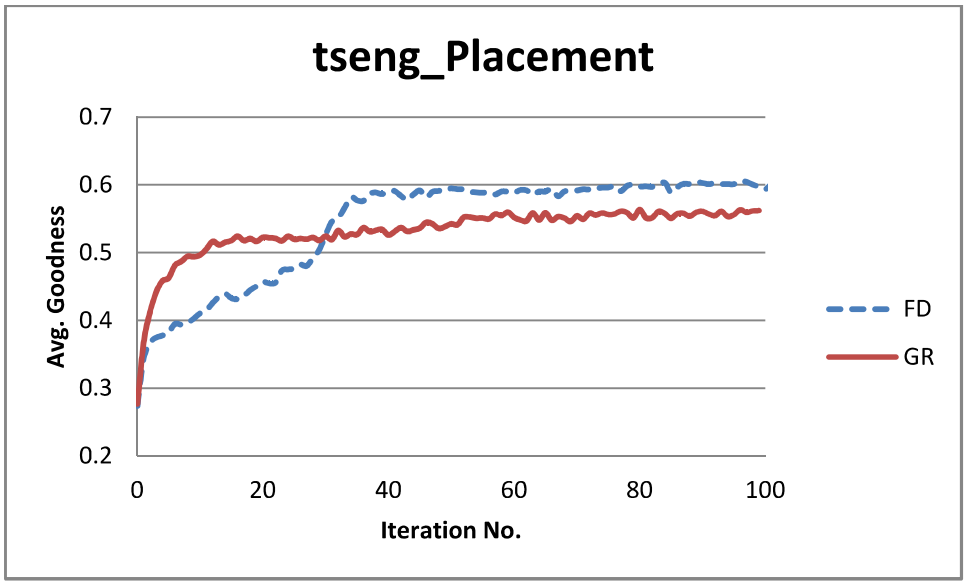

Figure 4.8 : Change in Avg. Goodness of tseng circuit in the Placement Problem with Iteration.

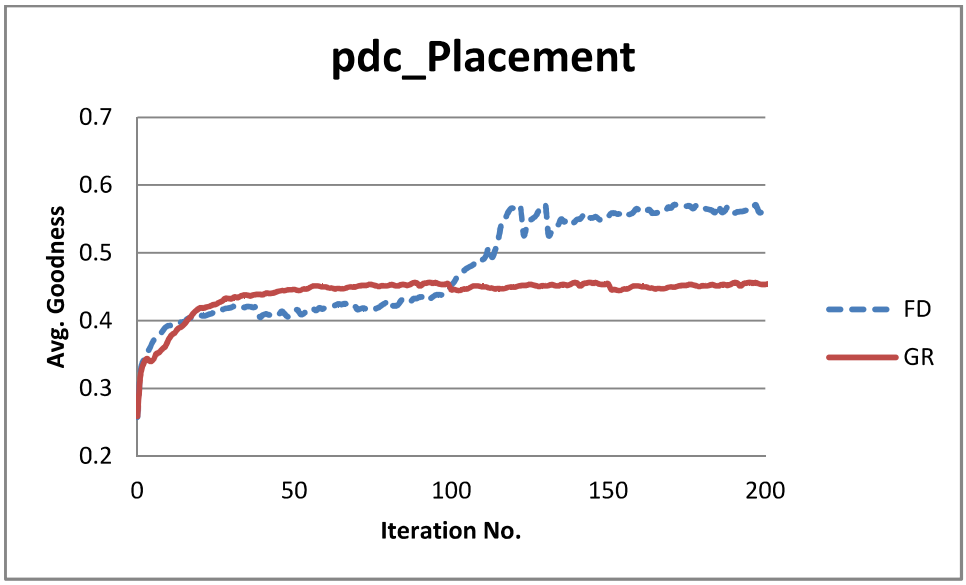

Figure 4.9: Change in Avg. Goodness of pdc circuit in the Placement Problem with Iteration.

Figures 4.12 and 4.13 show the WL vs. iteration plots for the same circuits. As we can see from the figures, at the early iterations the WL values are large but after only small number of iterations the WL value reduced significantly and converges to the desired final result. Also we can observe that the FD approach always gives the minimum WL, while the GR improves the WL slowly in the later iterations.

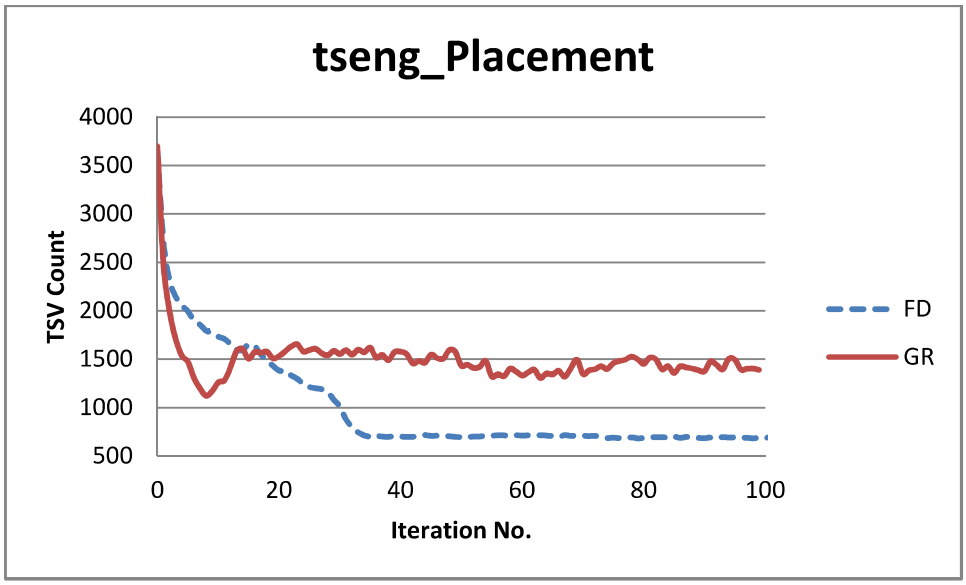

Figure 4.10: Change in TSV Count of tseng circuit in the Placement Problem with Iteration.

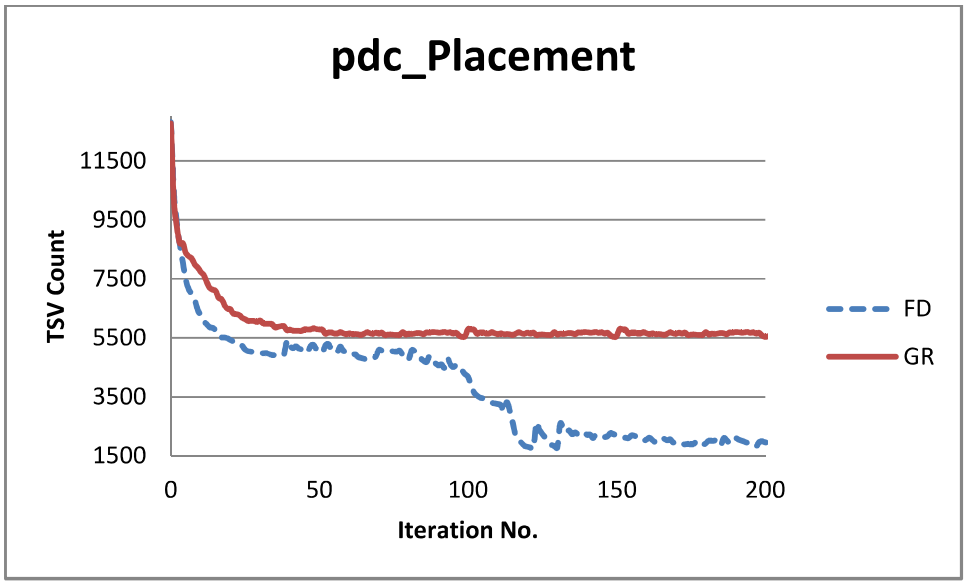

Figure 4.11: Change in TSV Count of pdc circuit in the Placement Problem with Iteration.

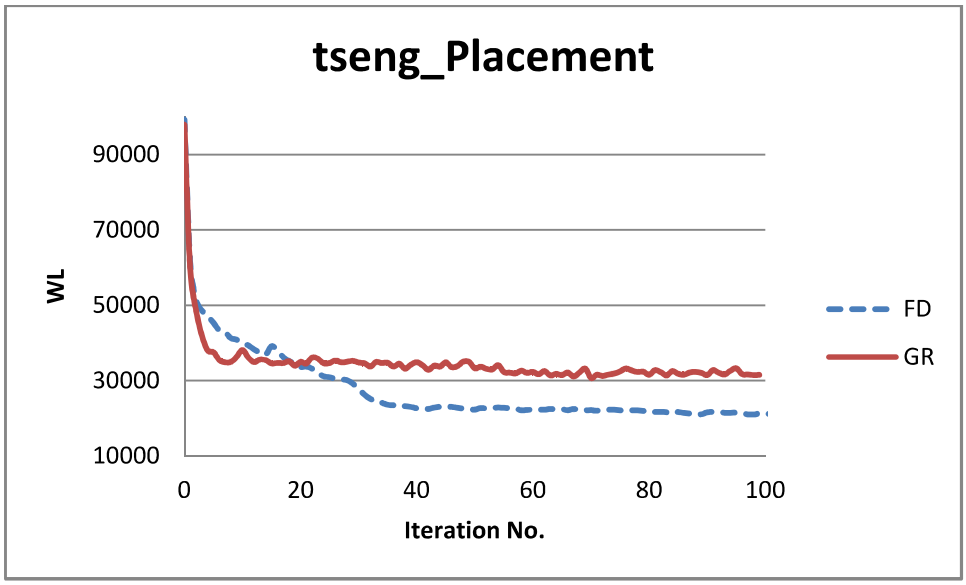

Figure 4.12: Change in WL of tseng circuit in the Placement Problem with Iteration.

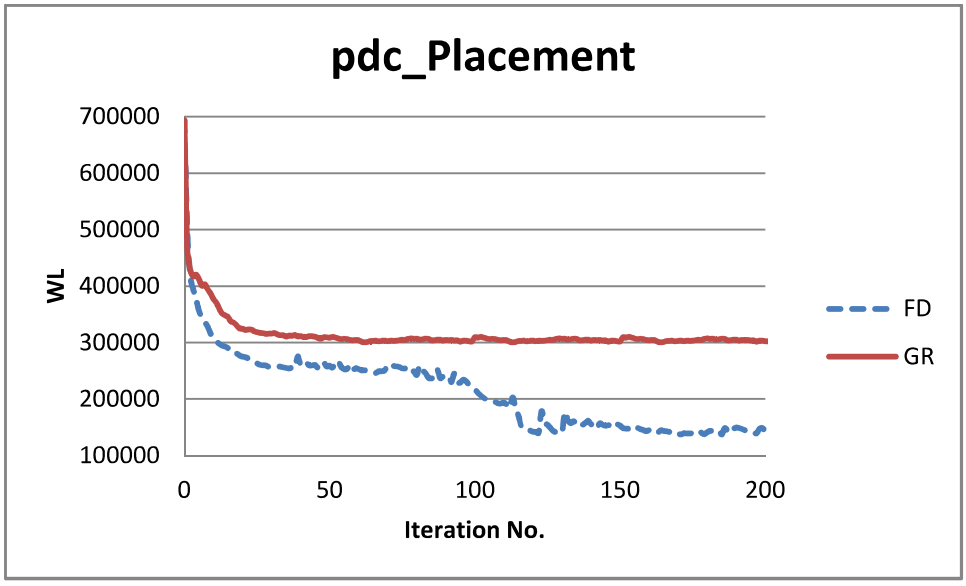

Figure 4.13: Change in WL of pdc circuit in the Placement Problem with Iteration.

The Combined objective (TCO) of both circuits is shown in Figures 4.14 and 4.15. As we can notice in the figures, the TCO initially is low and this is because the higher values of the TSV Count and the overall WL; as the algorithm proceeds, the TSV Count and WL decrease quickly and consequently the TCO is improved after small number of iterations. Also we can observe that the FD approach always gives the highest TCO, while the GR improves the TCO slowly in the later iterations.

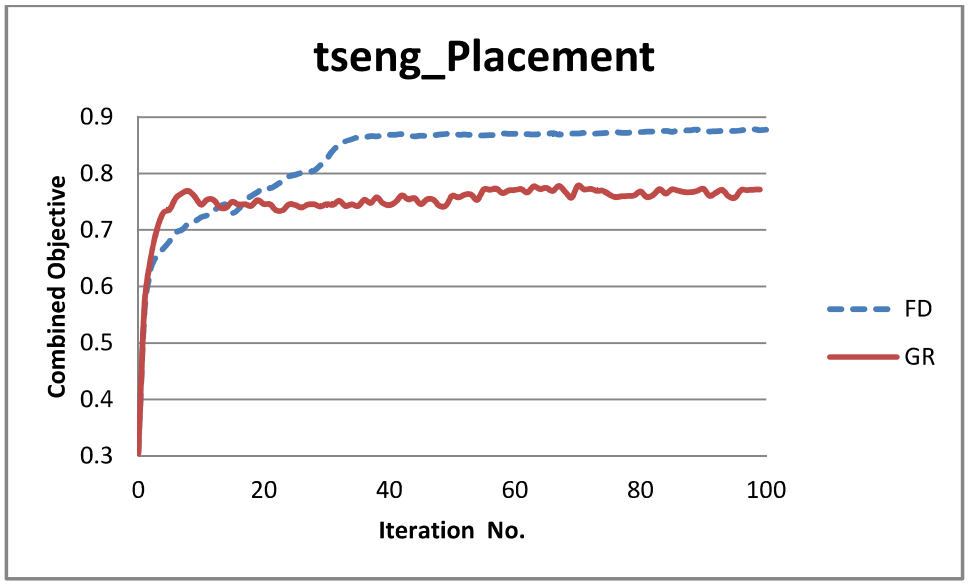

Figure 4.14: Change in TCO of tseng circuit in the Placement Problem with Iteration.

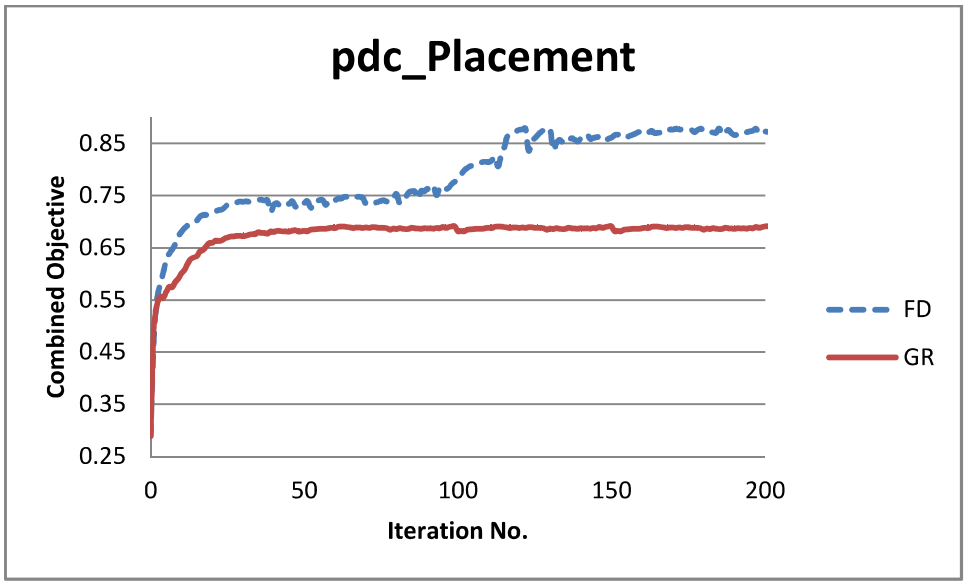

Figure 4.15: Change in TCO of pdc circuit in the Placement Problem with Iteration.

CHAPTER 5

CONCLUSION AND FUTURE WORK

3-D IC is a recent powerful technology that improves the VLSI design quality especially the interconnect delay requirements and the chip flat area. In this technology, multiple dies are stacked vertically and connected by a special 3-D vias known as TSV. A TSV occupy a significant silicon area when compared to the gates and modules, and because of that a special methods are needed to handle these vias during the physical design process. The number of TSVs and the distribution of them are the two main challenges that are imposed on 3-D IC design. In this work we studied this new technology along with the TSV, and we investigated the state representations and objective functions which can be effectively used in optimizing different 3-D IC physical design problems.

This work utilizes different well-known iterative heuristics to optimize two important issues in the 3D physical design process: the TSV minimization problem and the cell placement problem. The first part of this work handles the TSV minimization problem taking into account the layers area balancing constraint. To solve this problem we deployed a modified SA algorithm (GBSA) that incorporates a goodness based selection in the standard SA. The results show that this algorithm outperforms the standard SA algorithm in terms of the solution quality and the execution time. Also we applied the SimE algorithm to solve the TSV optimization problem and the results highlight the strength of the SimE approach that outperforms the SA, the modified SA (GBSA) and the TS algorithms.

The second part of this work handles the placement problem in 3D cell design. To our knowledge no one has solved this problem considering the same objectives and assumptions that we have considered. This problem targets the optimization of the TSV count and the overall wirelength. To solve this hard problem we applied the SimE algorithm using different allocation schemes. One of these allocation methods is the well-

known Force Directed (FD) method. Utilizing the FD method in a well-defined and adapted SimE algorithm produces high quality placement with minimum number of TSVs and reduced wirelength.

As a next step, the researchers can explore more goodness measures and allocation methods that can improve the solution quality or speed up the execution time of the developed approaches. Also in the placement problem, further researches could apply the SimE algorithm with the consideration of that the size of TSV is much larger than the modules in the 3D cell design. Another issue that needs to be handled in the future is the deployment of iterative algorithms to solve other 3D physical design issues such as the floorplanning problem.

REFERENCES

[1] I. Roadmap. "International Technology Roadmap for Semiconductors, 2009 edn." *Executive Summary. Semiconductor Industry Association* (2009).

[2] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat. "3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration." *Proceedings of the IEEE* 89, no. 5 (2001): 602-33.

[3] A. W. Topol, D. La Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar*, et al.* "Three-dimensional integrated circuits." *IBM Journal of Research and Development* 50, no. 4.5 (2006): 491-506.

[4] P. Garrou, C. Bower, and P. Ramm. *Handbook of 3D Integration: Volume 1-Technology and Applications of 3D Integrated Circuits*. John Wiley & Sons, 2011.

[5] C.-R. Li, W.-K. Mak, and T.-C. Wang. "Fast fixed-outline 3-D IC floorplanning with TSV co-placement." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, no. 3 (2013): 523-32.

[6] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, and P. Wyatt. "Three-dimensional integrated circuits for low-power, high-bandwidth systems on a chip." IEEE International Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. , 2001.

[7] D. Noice, and V. Gerousis. "Physical design implementation for 3D IC– Methodology and tools." Invited talk at Intl. Symposium on Physical Design, 2010.

[8] M.-C. Tsai, T.-C. Wang, and T. Hwang. "Through-silicon via planning in 3-D floorplanning." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19, no. 8 (2011): 1448-57.

[9] L. Hou, S. Bai, and J. Wang. "Research on TSV positioning in 3D IC placement." IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), 2011.

[10] J. Baliga. "Chips go vertical [3D IC interconnection]." *IEEE Spectrum* 41, no. 3 (2004): 43-47.

[11] E. Beyne, P. De Moor, W. Ruythooren, R. Labie, A. Jourdain, H. Tilmans, D. S. Tezcan*, et al.* "Through-silicon via and die stacking technologies for microsystems-integration." IEEE International Electron Devices Meeting, 2008.

[12] M. Koyanagi, T. Fukushima, and T. Tanaka. "High-density through silicon vias for 3-D LSIs." *Proceedings of the IEEE* 97, no. 1 (2009): 49-59.

[13] H. Li, E. Liao, X. Pang, H. Yu, X. Yu, and J. Sun. "Fast electroplating TSV process development for the via-last approach." Proceedings of the 60th Electronic Components and Technology Conference (ECTC). , 2010.

[14] C. Chiang, and S. Sinha. "The road to 3D EDA tool readiness." Asia and South Pacific Design Automation Conference, 2009.

[15] S. M. Sait, and H. Youssef. *VISI Physical Design Automation: Theory and Practice*. McGraw-Hill, Inc., 1994.

[16] S. M. Sait, and H. Youssef. *Iterative computer algorithms with applications in engineering: solving combinatorial optimization problems*. IEEE Computer Society Press, 1999.

[17] D. B. Fogel. "An introduction to simulated evolutionary optimization." *IEEE Transactions on Neural Networks* 5, no. 1 (1994): 3-14.

[18] R. M. Kling, and P. Banerjee. "Esp: Placement by simulated evolution." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 8, no. 3 (1989): 245-56.

[19] M. Hanan, and J. M. Kurtzberg. "A review of the placement and quadratic assignment problems." *Siam Review* 14, no. 2 (1972): 324-42.

[20] L. A. Zadeh. "Fuzzy sets." *Information and control* 8, no. 3 (1965): 338-53.

[21] W. Zhong, S. Chen, and T. Yoshimura. "Whitespace insertion for through-silicon via planning on 3-D SoCs." Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), 2010.

[22] Y.-S. Huang, Y.-H. Liu, and J.-D. Huang. "Layer-Aware Design Partitioning for Vertical Interconnect Minimization." IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2011.

[23] M.-K. Hsu, Y.-W. Chang, and V. Balabanov. "TSV-aware analytical placement for 3D IC designs." Design Automation Conference (DAC), 2011.

[24] M.-K. Hsu, V. Balabanov, and Y.-W. Chang. "TSV-Aware Analytical Placement for 3-D IC Designs Based on a Novel Weighted-Average Wirelength Model." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, no. 4 (2013): 497-509.

[25] J. Cong, G. Luo, and Y. Shi. "Thermal-aware cell and through-silicon-via co-placement for 3D ICs." Proceedings of the 48th Design Automation Conference, 2011.

[26] M. Pathak, Y.-J. Lee, T. Moon, and S. K. Lim. "Through-silicon-via management during 3D physical design: When to add and how many?", Proceedings of the International Conference on Computer-Aided Design, 2010.

[27] B. Goplen, and S. Sapatnekar. "Efficient thermal placement of standard cells in 3D ICs using a force directed approach." Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design, 2003.

[28] B. Goplen, and S. Spatnekar. "Placement of 3D ICs with thermal and interlayer via considerations." Proceedings of the 44th annual Design Automation Conference, 2007.

[29] J. Cong, and G. Luo. "A multilevel analytical placement for 3D ICs." Proceedings of the 2009 Asia and South Pacific Design Automation Conference, 2009.

[30] Z. Li, X. Hong, Q. Zhou, Y. Cai, J. Bian, H. H. Yang, V. Pitchumani, and C.-K. Cheng. "Hierarchical 3-D floorplanning algorithm for wirelength optimization." *IEEE Transactions on Circuits and Systems I: Regular Papers* 53, no. 12 (2006): 2637-46.

[31] V. F. Pavlidis, and E. G. Friedman. "Interconnect-based design methodologies for three-dimensional integrated circuits." *Proceedings of the IEEE* 97, no. 1 (2009): 123-40.

[32] Y. C. Hu, Y. L. Chung, and M. C. Chi. "A multilevel multilayer partitioning algorithm for three dimensional integrated circuits." 11th International Symposium on Quality Electronic Design (ISQED), 2010.

[33] D. H. Kim, K. Athikulwongse, and S. K. Lim. "Study of Through-Silicon-Via Impact on the 3-D Stacked IC Layout." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, no. 5 (2013): 862-74.

[34] I.-R. Jiang. "Generic integer linear programming formulation for 3D IC partitioning." IEEE International SOC Conference, 2009.

[35] S. M. Sait, F. Oughali, and M. Alasli. "Design Partitioning and Layer Assignment for 3D Integrated Circuits Using Tabu Search and Simulated Annealing." 2014.

[36] X. He, S. Dong, Y. Ma, and X. Hong. "Simultaneous buffer and interlayer via planning for 3D floorplanning." Quality of Electronic Design, 2009.

[37] S. N. Adya, and I. L. Markov. "Fixed-outline floorplanning: Enabling hierarchical design." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 11, no. 6 (2003): 1120-35.

[38] S. Chen, and T. Yoshimura. "Multi-layer floorplanning for stacked ICs: Configuration number and fixed-outline constraints." *The VLSI journal INTEGRATION* 43, no. 4 (2010): 378-88.

[39] J. Lu, S. Chen, and T. Yoshimura. "Performance maximized interlayer via planning for 3D ICs." 7th International Conference on ASIC, 2007.

[40] D. H. Kim, K. Athikulwongse, and S. K. Lim. "A study of Through-Silicon-Via impact on the 3D stacked IC layout." Proceedings of the 2009 International Conference on Computer-Aided Design, 2009.

[41] T. Zhang, Y. Zhan, and S. S. Sapatnekar. "Temperature-aware routing in 3D ICs." Asia and South Pacific Conference on Design Automation 2006.

[42] X. Li, Y. Ma, X. Hong, S. Dong, and J. Cong. "LP based white space redistribution for thermal via planning and performance optimization in 3D ICs." Asia and South Pacific Design Automation Conference 2008.

[43] J. Cong, J. Wei, and Y. Zhang. "A thermal-driven floorplanning algorithm for 3D ICs." IEEE/ACM International Conference on Computer Aided Design, 2004.

[44] J. Cong, and Y. Zhang. "Thermal via planning for 3-D ICs." IEEE/ACM International Conference on Computer-Aided Design, 2005.

[45] Z. Li, X. Hong, Q. Zhou, S. Zeng, J. Bian, H. Yang, V. Pitchumani, and C.-K. Cheng. "Integrating dynamic thermal via planning with 3D floorplanning

algorithm." Proceedings of the 2006 international symposium on Physical design, 2006.

[46] J. Cong, G. Luo, J. Wei, and Y. Zhang. "Thermal-aware 3D IC placement via transformation." Asia and South Pacific Design Automation Conference, 2007.

[47] J. Li. "Post-placement thermal via planning for 3D integrated circuit." IEEE Asia Pacific Conference on Circuits and Systems, 2006.

[48] E. Wong, and S. K. Lim. "3D floorplanning with thermal vias." Proceedings of Design, Automation and Test in Europe, 2006.

[49] L. Xiao, S. Sinha, J. Xu, and E. F. Young. "Fixed-outline thermal-aware 3D floorplanning." 15th Asia and South Pacific Design Automation Conference (ASP-DAC) 2010.

[50] H. Yan, Z. Li, Q. Zhou, and X. Hong. "Via assignment algorithm for hierarchical 3d placement." Proceedings of International Conference on Communications, Circuits and Systems, 2005. , 2005.

[51] X. Liu, G. Yeap, J. Tao, and X. Zeng. "Integrated Algorithm for 3-D IC Through-Silicon Via Assignment." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22, no. 3 (2014): 655-66.

[52] X. Liu, Y. Zhang, G. Yeap, and X. Zeng. "An integrated algorithm for 3D-IC TSV assignment." Proceedings of the 48th Design Automation Conference, 2011.

[53] K. Bazargan, R. Kastner, and M. Sarrafzadeh. "3-D floorplanning: simulated annealing and greedy placement methods for reconfigurable computing systems." IEEE International Workshop on Rapid System Prototyping 1999.

[54] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu. "Corner block list: an effective and efficient topological representation of non-slicing floorplan." IEEE/ACM International Conference on Computer Aided Design, 2000.

[55] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou. "3D-STAF: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits." IEEE/ACM International Conference on Computer-Aided Design, 2007.

[56] D. A. Papa, and I. L. Markov. "Hypergraph partitioning and clustering." *Approximation algorithms and metaheuristics* (2007): 61.1-61.19.

[57] K. Hayes. "On the Attainability of Bounds on the Standard Deviation." *Teaching Statistics* 32, no. 2 (2010): 54-56.

Vitae

ENG. MOHAMMED AHMAD ABDEL RAHMAN KHALIL

Saudi Arabia, Dhahran eng.m.khalil87@gmail.com (+966) 50-393-1089

PERSONAL INFORMATION

- **Marital status**: Single.
- **Nationality**: Jordanian.
- **Date of Birth**: Apr. 27th . 1987.
- **Place of Birth**: Doha – Qatar.

EDUCATIONAL BACKGROUND

- **M.Sc Degree**

  Period: 2012 - 2014.
  University: KFUPM.
  Program of study: Computer Engineering.
  Research Interest: Computer Hardware and Design Automation.
  GPA: 3.96 out of 4

- **B.S. Degree**

  Period: 2006 - 2011.
  University: Birzeit University, Palestine.
  Program of study: Computer Systems Engineering.
  Cumulative Average: 89% (*High Distinction*).
  Cumulative Average in major: 89%.

- **General Secondary School Certificate**

  Year: 2005.
  Branch: Scientific.
  Grade average: 89.3%.
  School: Al-Imam Ali Secondary Boys School – Amman – Jordan.

- **Papers & Patents**
  - *Sadiq* M. Sait, ***Mohammed A. Khalil*** , Feras Oughli "Multi-objective Optimization for 3D ICs Using Simulated Evolution" ***(submitted)***.

- *Feras M. Kafiah, Husam M. Walwil,* ***Mohammed A. Khalil****, Fadi M. Abu Samra* "Long lasting concrete curing blankets using super-absorption polymer" ***(Filed US Patent 14620139).***

WORK EXPERIENCE

- Birzeit University Jun-Jul 2009
  - **Technical Support**
    In this training I worked with a university team who is responsible for support the staff computers, maintain the university Servers, and maintain the university network and internet. Experience gained from this training was:
    - Installing and configuring Windows Server 2003.
    - Wired Networks setup, routers configuration, system administration.
    - Solving several PC Problems such as Firewall settings, LAN configuration.
- Birzeit University Jul-Aug 2009
  - **Wireless Networks Engineer**
    I worked with a networks engineer and another trainee on covering the university campus by wireless LAN. Experience gained from this training was:
    - Deep understanding of wireless networks design concepts.
    - Advanced knowledge about different available access points, and configuring them.
    - Advanced experience about wireless security protocols, and making decisions to choose the appropriate protocol based on case requirements.
- Freelancer, Jordan 2011-2012
  - Building and developing Integrated Systems, websites and desktop applications for individuals and small institutes and firms.
- Teaching and Assisting BS students, KFUPM 2012-2014
  - Conducting group sessions in Technology Solutions, physics, basic mathematics and English for Orientation Year Students.
  - Conducting supportive sessions in various IT and engineering areas for engineering students.

ACHIVMENTS & AWARDS

- 2nd Venture Concept Competition 2014, First Winners at KFUPM.
- 1st KFUPM Concept to prototype competition 2014, First winner.
- Participated in The Sixth Scientific Conference for Students of Higher Education in KSA, In the Innovation Track, 2015.
- Participated in The Sixth Scientific Conference for Students of Higher Education in KSA, In the Entrepreneurship Track, 2015.

- Organizer of the 1st and 2nd Computer Engineering Symposium, KFUPM.
- Scientific explainer for the 1001 invention at ARAMCO IETHRA Knowledge Exhibition 2013.
- Full Scholarship from (KFUPM) to study MS in Computer Engineering 2012.
- Belgian Grant for one semester Due to High GPA.
- I was on the university honor list for nine semesters.